Enhancing Prediction Efficacy with High-Dimensional Input Via Structural Mixture Modeling of Local Linear Mappings

by

Chun-Chen Tu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Statistics)
in The University of Michigan
2019

Doctoral Committee:

Senior Researcher Florence Forbes, Co-Chair
Professor Naisyin Wang, Co-Chair
Associate Professor Veronica Berrocal
Dr. Pin-Yu Chen
Professor Kerby Shedden

Chun-Chen Tu

timtu@umich.edu

ORCID iD: 0000-0002-6619-9521

© Chun-Chen Tu 2019

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

ABSTRACT

Regression is a widely used statistical tool to discover associations between variables. Estimated relationships can be further utilized for predicting new observations. Obtaining reliable prediction outcomes is a challenging task. When building a regression model, several difficulties such as high dimensionality in predictors, non-linearity of the associations and outliers could reduce the quality of results. Furthermore, the prediction error increases if the newly acquired data is not processed carefully. In this dissertation, we aim at improving prediction performance by enhancing the model robustness at the training stage and duly handling the query data at the testing stage. We propose two methods to build robust models. One focuses on adopting a parsimonious model to limit the number of parameters and a refinement technique to enhance model robustness. We design the procedure to be carried out on parallel systems and further extend their ability to handle complex and large-scale datasets. The other method restricts the parameter space to avoid the singularity issue and takes up trimming techniques to limit the influence of outlying observations. We build both approaches by using the mixture-modeling principle to accommodate data heterogeneity without uncontrollably increasing model complexity. The proposed procedures for suitably choosing tuning parameters further enhance the ability to determine the sizes of the models according to the richness of the available data. Both methods show their ability to improve prediction performance, compared to existing approaches, in applications such as magnetic resonance vascular fingerprinting and source separation in single-channel polyphonic music, among others. To evaluate model robustness, we develop an efficient approach to generating adversarial samples,

which could induce large prediction errors yet are difficult to detect visually. Finally, we propose a preprocessing system to detect and repair different kinds of abnormal testing samples for prediction efficacy, when testing samples are either corrupted or adversarially perturbed.

CHAPTERS I

Introduction

1.1 Motivation

Regression is a widely used statistical tool in all disciplines. One primary goal of building regression models is to conduct prediction. Given pairs of covariates and responses, we aim to find the relation between the covariates and responses so that if a new covariate is observed, one can predict the corresponding response. As an example, researchers in modern geosciences are interested in recovering physical parameters using hyperspectral images (Bioucas-Dias et al., 2012; Brown et al., 2000). Applications of remote sensing include monitoring earthquakes or tracking chemical contamination. Many of the applications of remote sensing require accurate prediction results. As natural disasters occur, decisions are made based on the prediction results and thus if the outcomes are not trustworthy, inappropriate decisions could incur more loss. A similar application can be found in the field of magnetic resonance imaging. The goal is to effectively assess microvascular properties such as blood volume fraction, vessel diameter and blood oxygenation (Lemasson et al., 2016) to improve diagnosis of brain diseases. Magnetic resonance imaging can assist in brain disease diagnosis. However, professionals could misjudge a condition if unreliable results are provided.

Obtaining reliable prediction outcomes is a challenging task. Plenty of factors

influence the performance of a regression model, including but not limited to outliers, high dimensionality, and non-linearity. Moreover, even if when a robust model is built, one can still obtain unreliable prediction outcomes if the newly observed data are not carefully processed. For instance, testing data could be corrupted or distorted. For remote sensing, unknown human-made objects could lead to unexpected prediction behavior. Noisy samples are commonly seen when data are collected in vivo. These are abnormal data that could naturally appear when conducting prediction during the testing stage. On the other hand, malicious inputs can be designed to make a model fail. These artificially designed inputs can be seen in spam filtering (Fawcett, 2003), fraud detection (Mahoney and Chan, 2002) and object detection (Song et al., 2018).

In this dissertation, we are interested in obtaining reliable prediction outcomes. This goal can be achieved through robust learning and prediction at the training and testing stages. In particular, we focus on building robust regression models for prediction given a dataset with a complicated structure. The model should be capable of learning non-linear structures under a high-dimensional setting and should be robust against the presence of outliers. For evaluating model robustness at the testing stage, we devise a method for generating abnormal data efficiently. The proposed method is a general algorithm that can be applied to a rich class of models. Moreover, we propose a preprocessing system to cope with abnormal testing data. The system is applied to existing predictive methods, and we show that with the preprocessing system, the prediction performance can be improved.

1.2 Research challenges

Building a robust regression model on real-world datasets is a vexing problem. For one thing, the curse of dimensionality could severely influence model performance; for

another, the presence of outliers could lead to unreliable model outcomes; lastly, the data may possess sophisticated associations between covariates and responses, which could make it difficult to strike a balance between model complexity and model capability. For the training stage, we focus on three commonly seen issues when building regression models on real-world datasets – namely, high dimensionality, outliers and non-linearity between covariates and responses.

Training with high-dimensional covariates

High-dimensional data analysis has drawn great attention as a consequence of rapid advancement in technology. Data have become easier to collect, but associations between high-dimensional covariates and low-dimensional responses are more difficult to analyze. When building models with high-dimensional predictors, we often encounter situations in which the number of parameters is larger than the sample size. This leads to unstable estimation and might reduce the quality of the model performance. Even worse, if data are correlated, it is necessary to estimate covariance matrices for model-based methods. The covariance matrices estimated in this situation are not full-rank, which leads to numerical issues when there is a need to calculate the inverse of the covariance matrices. A common practice is to reduce the dimensionality of the data and then perform a regression analysis on the reduced space. As an example, principal component regression (PCR) adopts principal component analysis (PCA) for dimension reduction. The regression analysis is then conducted on the responses and the PCA loadings. PCR can effectively reduce the dimension of the predictors' space to overcome the difficulty of high dimensionality. However, PCR excludes principal components with low variance, which may be critical to conduct the regression analysis (Jolliffe, 1982). Partial least squares (PLS) (Turkmen and Billor, 2013) treats covariates and responses as two separate systems and decomposes these two systems into matrices with score vectors multiplied by matrices with loadings.

The regression mapping is then estimated by finding the strongest linear relationship between two score vectors. The performance of PLS depends on the relation between the covariances of covariates and responses. Thus, if there is no strong associations between the covariances of covariates and responses, one would experience reduction in the model quality. Sliced inverse regression (SIR, Li 1991) exchanges the role of covariates and responses. It is designed to find the dimension reduction space and the results highly depend on the selection of slices. In addition, the method is not designed for prediction and thus the solution could be sub-optimal.

Sparsity assumption is another class of methods to overcome high dimensionality (Städler et al., 2010; Tibshirani, 1996; Yi and Caramanis, 2015). This category of methods considers the situation that not every predictor is informative for model construction. The presence of non-informative predictors might increase the model variance and worsen its predictive performance. When building the model, this type of methods only uses a subset of the predictors. This can be done by adding an extra L_1 penalty term to the objective function. The penalty term would suppress the value of the regression coefficients, which makes some of the regression coefficients become zeroes. Zero regression coefficients imply that the corresponding covariates have no contribution when conducting prediction. The prediction variance can be effectively reduced by ruling out non-informative covariates.

With the increasing necessity to handle high-dimensional settings, parsimonious models have gained much attention in recent years. Parsimonious models generally refer to some model instances where the number of parameters is reduced compared to the full parameterization. When dealing with high-dimensional data, the goal is to find a good compromise between model flexibility and parsimony. Complex models cannot be estimated accurately in most real-life settings because of data insufficiency, and simple models may not be able to capture the full data structure. In terms of the number of parameters, the largest costs usually come from high-dimensional covari-

ance matrices. Diagonal covariance matrices are parsimonious and are often tractable in high dimensions. However, they cannot capture complex dependence structures. Significant gains are then expected from adopting parsimonious covariance representations. In model-based clustering (Banfield and Raftery, 1993; Fraley and Raftery, 2002), covariance matrices are decomposed using eigenvalues and eigenvectos. The number of parameters can be reduced by assuming constraints on the eigendecomposition. Factor models (McLachlan and Peel, 2000) that induce a decomposition into a diagonal and low-rank matrix also result in a parsimonious parameterization. In a mixture of regressions context, the work of Subedi et al. (2013) uses a cluster-weighted factor analyzer (CWFA) approach when the number of covariates is large. The high dimensionality issue is overcome by imposing constraints on the covariance matrix of the high-dimensional variables. Gaussian Locally Linear Mapping (GLLiM) (Deleforge et al., 2015) also adopts a factor model similar to parameterization for high-dimensional covariance matrices but with a different interpretation.

Training with non-linear associations

Linear regression is a common statistical approach to establish the relationship between covariates and responses. However, the assumption of linearity may not always be valid. Especially when conducting regression analysis on real-world datasets, the associations between covariates and responses are often beyond linear. To properly model complicated associations while maintaining tractability, non-linear mappings can be handled through a transformation of the data into a so-called feature space using kernel methods. For example, kernel SIR (Wu, 2008) proposed a hybrid SIR together with kernel methods to deal with non-linear data. Kernel SIR is a powerful tool for dimension reduction and feature extraction for non-linear data. Other examples of kernel methods include Elisseeff and Weston (2002), Lawrence (2005) and Thayananthan et al. (2006) to name a few. All kernel methods share a common issue

of how to select an appropriate kernel function. A proper kernel function can only be chosen heuristically and is usually data-dependent, which results in a time-consuming tuning process.

Another solution to non-linear data is to utilize a mixture of regression models (De Veaux, 1989; Frühwirth-Schnatter, 2006; Goldfeld and Quandt, 1973). Data are broken down into several clusters with each cluster following linear associations between covariates and responses. These clusters form a mixture, and thus the relationship learned in this manner is non-linear. The conventional mixture of regressions is not appropriate for regression because it assumes assignment independence (Hennig, 2000). This indicates that the assignments to each of the regression components are independent of the covariate values. In contrast, when a mixture of locally linear models is considered, one can let the membership indicator of the mixture component depend on the values of the covariates. Consequently, when extended with assignment dependence, models in the family of mixtures of regressions are more likely to be suitable for regression applications. In Baek et al. (2010), local linearity is used to group data with similar regression structures, and thus the number of parameters can be reduced through grouping. The Gaussian cluster-weighted (CW) model (Gershenfeld, 1997) also adopts mixtures to approximate non-linear associations. The CW model is extended with the factor analyzer approach (CWFA) (Subedi et al., 2013) to handle a large number of covariates. Qiao and Minematsu (2009) proposed a mixture of probabilistic linear regressions (MPLR) to learn the non-linear mappings. MPLR is claimed to be a more general framework than the methods based on Gaussian mixture model and can achieve better performance for the task of voice conversion. However, the parameter estimation procedure requires the sample size to be sufficiently large to estimate covariance matrices properly. Otherwise, numerical issues could arise when the dimension of the predictor is larger than the number of samples. GLLiM also adopts local linearity. With a carefully designed framework, the model parame-

ters can be efficiently estimated and the issue of high dimensionality can be bypassed.

Training with outliers and achieving stability

At the training stage, two factors could affect the robustness of the model: outliers and stability. Outliers are defined as abnormal data that do not follow the major pattern in the dataset. When estimating model parameters, traditional methods often target minimizing mean squared errors between the responses and the predicted values. If outliers exist, the squared errors corresponding to these outliers will be larger than those of other normal data. Thus, the estimation procedure tends to reduce the squared errors of outliers, which increases the squared errors of normal data. Methods like classical multiple linear regression and PLS are known to be sensitive to outliers.

One class of methods aims to alleviate the impact of outliers. Examples like Least Median of Squares (LMS) (Rousseeuw, 1984) replaces the least sum of squares with the least median of squares. Since the median cannot be influenced much by extreme values, LMS enhances model robustness against outliers. On the other hand, if a Gaussian distribution is used for the model-based method, the influence of outliers can be mitigated by using a Student's t distribution instead of a Gaussian distribution for the error terms (Archambeau and Verleysen, 2007; Peel and McLachlan, 2000; Perthame et al., 2018; Yao et al., 2014).

Another typical approach to handle outliers is to detect and trim them. Cook's distance (Cook, 1977) is a commonly used metric for detecting outliers. Though Cook's distance suffers from several drawbacks, it is widely used and implemented in a variety of applications. The trimming approach is adopted in Cuesta-Albertos et al. (1997) to remove abnormal samples with low likelihood of belonging to the current model, which enhances the robustness of k-means clustering when equality of variances is assumed. The trimming problem becomes much harder for heterogeneous data (García-Escudero and Gordaliza, 2007). To cope with the heterogeneous

problem, Gallegos and Ritter (2005) use the "spurious-outliers model" to define a likelihood function based on normal samples and abnormal samples. By maximizing the likelihood function specified in the spurious-outliers model, abnormal samples are naturally trimmed out. The estimation procedure is then performed on the remaining training samples and, as a consequence, is no affected by the influence of outliers. In Neykov et al. (2007), a similar trimmed likelihood framework is adopted. The trimmed likelihood estimator can robustly fit the mixture models and provide reliable estimates when abnormal data exist.

Model stability is a well-known issue for the likelihood-based approach in the mixture context. The likelihood function could be unbounded, which indicates that a global maximum likelihood estimate does not exist. This is also known as the singularity problem since the likelihood function approaches infinity at singular points. A large amount of research has been done to resolve the singularity issue. One solution is to add a penalty term to the covariance matrices (Ciuperca et al., 2003; Ridolfi and Idier, 2001; Snoussi and Mohammad-Djafari, 2002). In Chen and Tan (2009), the authors found that the penalized likelihood estimator is strongly consistent if the penalty function satisfies several conditions. Another method to tackle the singularity problem is to restrict the relative ratio between variances or covariance matrices. In Hathaway (1985), the relative ratio constraint is imposed on univariate Gaussian mixtures. The constrained likelihood is statistically well-posed and the strong consistency of the estimators is shown. The idea is extended in García-Escudero et al. (2008), where the constraint is applied to eigenvalues.

Generating adversarial data for robustness evaluation

A robust training process does not necessarily imply that one can obtain reliable prediction outcomes from the testing data. Traditional machine learning algorithms assume the data generating process is independent of the model. However, in many

real-world applications, this assumption is invalid. As an example, the performance of a spam filtering model can quickly downgrade after it is deployed. Spammers learn to insert non-spam words into emails to fool the filter. As the spam filter is updated with these tricks, spammers develop new techniques to bypass the filter (Brückner et al., 2012; Fawcett, 2003). These endless arms races occur in other fields as well, such as fraud detection (Fawcett and Provost, 1997), web search (Mahoney and Chan, 2002) and ad-blocking (Tramèr et al., 2018). These carefully crafted data are called adversarial data. This kind of data is intentionally generated to make the target model fail. Therefore, to evaluate the robustness of a prediction process, one should consider adversarial examples as a kind of possible input. Also, an efficient approach to generating adversarial examples is necessary to effectively assess model robustness. The generation of adversarial data is often formulated as an optimization problem. A loss function is carefully designed so that the optimizer of the function could cause model failure while fulfilling some constraints (e.g., the magnitude of perturbations is less than some upper bound so that it looks like its normal counterpart). Effective adversarial example generation methods include the fast gradient sign method (Kurakin et al., 2017), Carlini and Wagner's (C&W) attack (Carlini and Wagner, 2017) and the elastic-net attack (Chen et al., 2018). These methods assume that the model information is transparent when generating adversarial examples, which is often referred to as the "white-box" setting. On the other hand, researchers also consider the situation where the information is limited to the input-output relationship. The so-called "black-box" setting is more practical but more challenging. Under the "black-box" setting, methods like those proposed by Chen et al. (2017) and Nitin Bhagoji et al. (2018) estimate coordinate-wise gradients directly through model queries. However, to successfully generate adversarial examples, these methods require a large number of queries and thus are not efficient.

Improving prediction efficacy with abnormal inputs

The testing data could contain abnormal samples. If one blindly applies predictive models to aberrant data, the prediction efficacy could be reduced. To mitigate the influence of abnormal testing data, repair processes are necessary to make the problematic data points as close to the regular entries as possible. Data can be reconstructed from their closest representations in the latent space (Bhagoji et al., 2017; Elad and Aharon, 2006; Mairal et al., 2009). Since the latent space is learned from the regular samples, abnormal noises or corruptions would be eliminated from the reconstructed data. Nearest neighbors can also be utilized to restore the data. Barnes et al. (2009) proposed to use random sampling to search patch matches. These patches can be used to repair the corrupted parts. The task of image restoration can also be done by utilizing deep neural networks. Deep neural networks learn hidden representations and reconstruct images through convolutional filters (Pathak et al., 2016; Yang et al., 2017; Iizuka et al., 2017; Liu et al., 2018), which can produce much more meaningful repair results.

The repair processes need not be applied to regular data. To determine if repair is required, an extra detection stage is devised (Wang and Zhang, 1999; Roy et al., 2016). Outlier detection can be used in the detection procedure if the abnormal data possess visible irregular patterns. However, adversarial samples are usually similar to their normal counterparts. The differences between adversarial samples and normal ones can even be imperceptible. This makes detecting adversarial examples a challenging task. Grosse et al. (2017) adopted maximum mean discrepancy (MMD) (Gretton et al., 2012) for testing whether two sets of data are drawn from the same underlying distribution. However, bootstrapping is required for this method to approximate p-values, which makes the detection procedure computationally intensive. PCA whitening is utilized in Hendrycks and Gimpel (2017) to detect adversarial images. The coefficients of abnormal data after PCA whitening are different from those

of the normal ones, and can be utilized to conduct the detection.

1.3 Dissertation outline

This dissertation is organized as follows:

- Chapter II proposes a structured mixture model named Hierarchical Gaussian Locally Linear Mapping (HGLLiM) as a heuristic method for building models robustly. HGLLiM adopts a hierarchical structure which enables shared covariances and latent factors to limit the number of parameters. A robust estimation procedure is devised for model stability. Two moderate-size datasets are used to show the flexibility of HGLLiM as well as its robustness against outliers.

- Chapter III discusses the parallelization extension of HGLLiM. The parallelization technique can substantially reduce model building time. Two large-scale datasets are used to demonstrate the broad applicability of HGLLiM for handling complex data for prediction.

- Chapter IV establishes Robust-GLLiM (RGLLiM), which achieves the goal of robustness through trimming outliers and restricting relative cluster sizes. The devised Expectation-Maximization (EM) algorithm can nicely incorporate outlier trimming and cluster size controlling in an integrated framework. Studies on real-world datasets and simulations show that the model performance can be improved with regard to the robustness concern.

- Chapter V develops the Autoencoder-based Zeroth Order Optimization Method (AutoZOOM) for generating adversarial examples efficiently under black-box settings. AutoZOOM adopts dimension reduction and random gradient vector estimation to accelerate the generation process. Results of the image classification task illustrate that AutoZOOM is an efficient approach for assessing model

robustness.

- Chapter VI proposes a preprocessing system for obtaining reliable prediction results. The system contains an aberrant data detector to distinguish normal, corrupted and adversarial data, and an aberrant data corrector to modify problematic observations before conducting a prediction. Through reconstruction and nearest neighbor surrogates, prediction errors can be considerably reduced. The proposed system is a generic method that different predictive models can adopt. Results of three existing prediction methods illustrate the general usage of the proposed system.

CHAPTERS II

Hierarchical Gaussian Locally Linear Mapping

Building a regression model for prediction is widely used in all disciplines. A large number of applications consist of learning the association between responses and predictors and focusing on predicting responses for newly observed samples. In this work, we go beyond simple linear models and focus on predicting low-dimensional responses using high-dimensional covariates when the associations between responses and covariates are non-linear. Non-linear mappings can be handled through different techniques such as kernel methods (Elisseeff and Weston, 2002; Wu, 2008) or local linearity (De Veaux, 1989; Frühwirth-Schnatter, 2006; Goldfeld and Quandt, 1973). In general, conventional methods adopting local linearity assume assignment independence and are considered inadequate for regression (Hennig, 2000). Alternatively, one can utlilze a mixture-modeling strategy and let the membership indicator of a mixture component depend on the values of the covariates. Gaussian Locally Linear Mapping (GLLiM Deleforge et al., 2015) follows this principle.

GLLiM groups data with similar regression associations together. Within the same cluster, the association can be considered as locally linear, which can then be resolved under the classical linear regression setting. Besides adopting the framework of model-based clustering (Banfield and Raftery, 1993; Fraley and Raftery, 2002), GLLiM also takes on a factor-model based parameterization (Baek et al., 2010; Bou-

veyron et al., 2007; McLachlan and Peel, 2000; Xie et al., 2010) to accommodate high-dimensional and potentially dependent covariates (see Equation (2.10)). In particular, high-dimensional variables are postulated as a sum of two components: one that is linearly related to low-dimensional responses, and another which can be projected onto a factor model and then be presented as augmented latent variables. This data augmentation approach is applicable in many application scenarios, whenever certain variables are only partially observed or are corrupted with irrelevant information. The augmentation step, with added latent variables, acts as a factor analyzer modeling for the noise covariance matrix in the regression model. GLLiM is based on joint modeling of both responses and covariates, observed or latent. This joint modeling framework allows for the use of an inverse regression strategy to handle high-dimensional data.

However, when the covariate dimension is much higher than the response dimension, GLLiM may result in erroneous clusters at the low dimension, leading to potentially inaccurate predictions. Specifically, when the clustering is conducted at a high joint dimension, the distance at a low dimension between two members of the same cluster could remain large. As a result, a mixture component might contain several sub-clusters and/or outliers, violating the Gaussian assumption of the model. This results in a model misspecification effect that can seriously impact prediction performance. We demonstrate this phenomenon with a numerical example in Section 2.1. A natural way to lessen this effect is to increase the number of components in the mixture, making each linear mapping even more local. But this practice also increases the number of parameters to be estimated. Estimating parameters in a parsimonious manner is required to avoid over-parameterization. In addition, increasing the number of clusters could isolate some data points or lead to singular covariance matrices. Hence, a robust estimation procedure for model stability is also necessary.

In order to provide reliable prediction results, we propose a parsimonious approach

combined with a robust estimation procedure which we refer to as Hierarchical Gaussian Locally Linear Mapping (HGLLiM) to construct a stable model for predicting low-dimensional responses. HGLLiM inherits advantages from GLLiM for handling high-dimensional, non-linear regression with partially-latent variables. In terms of the number of parameters, the largest costs usually come from high-dimensional covariance matrices. On this front, HGLLiM follows a two-layer hierarchical clustering structure in which we reduce the number of covariance parameters in the model. HGLLiM also includes a pruning algorithm for eliminating outliers as well as determining an appropriate number of clusters. The number of clusters and training outliers determined by HGLLiM can be further used by GLLiM for improving prediction performance.

With the goal of investigating the flexibility in accommodating data structures and the ability to protect from influences of outliers, we evaluate our method on two datasets with different characteristics. The face dataset contains facial images, the associated angles of faces and the source of the light. There is no obvious cluster structure at first glance, nor the existence of real outliers. We use this dataset to evaluate the ability of HGLLiM to handle modeling regression through local linear approximations. The orange juice dataset contains continuous spectrum predictors and some abnormal observations. Using this dataset, we aim to show that HGLLiM is robust and can effectively identify outlying observations. We use these two moderate-size datasets to demonstrate how the method works on data with different features and demonstrate the insensitivity of tuning parameter selection in a wide range of selection domains. The analyses of two large-scale complex datasets using the parallelization technique can be seen in Chapter III.

We first start by introducing the framework of GLLiM in Section 2.1 and discuss the issue of GLLiM in the high-dimensional setting using the face dataset an example. The details of HGLLiM are presented in Section 2.2. The experimental results for

two real datasets are provided in Section 2.4. Finally, Section 2.5 concludes this work with a discussion.

2.1 Limitations of Gaussian Locally Linear Mapping (GLLiM)

Gaussian Locally Linear Mapping (GLLiM) can be boiled down to a joint Gaussian mixture model of both responses and covariates. It can be viewed as an affine instance of mixture of experts as formulated in Xu et al. (1995) or as a Gaussian cluster-weighted (CW) model (Gershenfeld, 1997) with multivariate responses. GLLiM adopts an inverse regression technique to overcome high dimensionality and approximate non-linear associations through Gaussian mixtures. Consider modeling the association between $Y \in \mathbb{R}^L$ and $X \in \mathbb{R}^D$ where $D \gg L$. The ultimate goal is to predict Y given observed X; however, GLLiM begins with estimating the mapping from Y to X. We assume a linear relationship between X and Y within a given cluster. Denote a missing variable Z as the cluster indicator such that if $Z = k$ then $X = A_k Y + b_k + E_k$, where matrix $A_k \in \mathbb{R}^{D\times L}$ and vector $b \in \mathbb{R}^L$ define the mapping from Y to X and E_k is the error term following Gaussian distribution with zero mean and covariance matrix $\Sigma_k \in \mathbb{R}^{D\times D}$. The joint density of (X, Y) can be decomposed in inverse direction and forward direction:

$$p(X = x, Y = y; \theta) = \sum_{k=1}^{K} p(X = x|Y = y, Z = k; \theta)p(Y = y|Z = k; \theta)p(Z = k; \theta) \tag{2.1}$$

$$= \sum_{k=1}^{K} p(Y = y|X = x, Z = k; \theta^*)p(X = x|Z = k; \theta^*)p(Z = k; \theta^*) \tag{2.2}$$

Equation (2.1) decomposes the joint density using the inverse regression relationship from low-dimensional data (Y) to high-dimensional data (X). In contrast, the

forward regression relationship from high-dimensional data (X) to low-dimensional data (Y) is expressed in Equation (2.2). With the assumption of Gaussianity, the structure for the inverse direction is defined by:

$$p(X = x|Y = y, Z = k; \theta) = \mathcal{N}(x; A_k y + b_k, \Sigma_k). \tag{2.3}$$

$$p(Y = y|Z = k; \theta) = \mathcal{N}(y; c_k, \Gamma_k), \tag{2.4}$$

$$p(Z = k; \theta) = \pi_k,$$

where $\theta = \{c_k, \Gamma_k, \pi_k, A_k, b_k, \Sigma_k\}_{k=1}^K$ is the collection of the inverse model parameters with K mixtures. The equality between Equation (2.1) and Equation (2.2) addresses the relationship between θ and the forward model parameter $\theta^* = \{c_k^*, \Gamma_k^*, \pi_k^*, A_k^*, b_k^*, \Sigma_k^*\}_{k=1}^K$. The relationship between θ and θ^* is described as follows:

$$\begin{aligned}
c_k^* &= A_k c_k + b_k, \\
\Gamma_k^* &= \Sigma_k + A_k \Gamma_k A_k^\top, \\
\pi_k^* &= \pi_k, \\
A_k^* &= \Sigma_k^* A_k^\top \Sigma_k^{-1}, \\
b_k^* &= \Sigma_k^* (\Gamma_k^{-1} c_k - A_k^\top \Sigma_k^{-1} b_k), \\
\Sigma_k^* &= (\Gamma_k^{-1} + A_k^\top \Sigma_k^{-1} A_k)^{-1}.
\end{aligned}$$

Finally, the prediction is done by:

$$\mathbb{E}[Y|X = x] = \sum_{k=1}^{K} \frac{\pi_k^* N(x; c_k^*, \Gamma_k^*)}{\sum_{j=1}^K \pi_j^* N(x; c_j^*, \Gamma_j^*)} (A_k^* x + b_k^*) \tag{2.5}$$

One feature of GLLiM is that Y need not be completely observable. That is, Y

can be decomposed as observed component T and latent component W,

$$Y = \begin{bmatrix} T \\ W \end{bmatrix}, \tag{2.6}$$

where $T \in \mathbb{R}^{L_t}$, $W \in \mathbb{R}^{L_w}$ and $L_w + L_t = L$. Assuming that T and W are independent given Z, we have

$$c_k = \begin{bmatrix} c_k^t \\ c_k^w \end{bmatrix}, \Gamma_k = \begin{bmatrix} \Gamma_k^t & 0 \\ 0 & \Gamma_k^w \end{bmatrix} \text{ and } A_k = \begin{bmatrix} A_k^t, A_k^w \end{bmatrix} \tag{2.7}$$

It follows that, for a given cluster k, we can rewrite the relationship between X and Y as:

$$X = A_k^t T + A_k^w W + b_k + E_k \tag{2.8}$$

$$= A_k^t T + b_k + A_k^w c_k^w + E_k', \tag{2.9}$$

where $E_k' \in \mathbb{R}^{D \times D}$ follows a Gaussian distribution with zero mean and covariance matrix:

$$\Sigma_k' = \Sigma_k + A_k^w \Gamma_k^w A_k^{w\top}. \tag{2.10}$$

The high-dimensional covariance matrix is modeled by the sum of the regression error from T to X and the component formed by latent variables. When Σ_k is diagonal, this structure is that of factor analysis with at most L_w factors, and represents a flexible compromise between a full covariance matrix with $O(D^2)$ parameters on one side, and a diagonal covariance matrix with $O(D)$ parameters on the other.

The issue with GLLiM is that the high dimensionality of the data may have an unexpected impact on the posterior probability of the cluster assignment. When the

dimensions of X and Y satisfy $D \gg L$, this comes from the following observation: in the E-step, the posterior probabilities of sample n belonging to cluster k, r_{nk} (Equation (27) in Deleforge et al., 2015) is computed as:

$$r_{nk} = p(Z_n = k|x_n, y_n; \theta) = \frac{\pi_k p(x_n, y_n|Z_n = k; \theta)}{\sum_{j=1}^{K} \pi_j p(x_n, y_n|Z_n = j; \theta)} \tag{2.11}$$

for all n and all k, where $p(x_n, y_n|Z_n = k; \theta)$ can be computed as $p(x_n|y_n, Z_n = k; \theta)p(y_n|Z_n = k; \theta)$. The first term is a density with a much higher dimension (D) so that its value could dominate the product. In addition, y_n can be decomposed into two parts: the observed variable t_n and the latent variable w_n. The component w_n reflects the remaining variation in x_n that cannot be explained by x_n's association with t_n. When w_n accounts for explaining most of the variation in x_n, the clustering outcome would highly depend on w_n and weaken the ability to detect sub-clusters in T.

Therefore, although GLLiM assumes that within each cluster $p(Y = y|Z = k; \theta)$ is Gaussian and centered at c_k, in practice, the model groups data according to the high-dimensional term and could fail to impose the Gaussian shape on the t_n's. In other words, the model rather chooses the clusters to satisfy the assumption in Equation (2.3). And this induces clustering of the (x_n, y_n)'s into groups within which the same affine transformation holds. Thus, a cluster could contain several sub-clusters and/or outliers since the Gaussian assumption on T, as part of the Y, in Equation (2.4) is sometimes neglected. This may cause a serious impact on the estimation of c_k and Γ_k, and consequently on the prediction step.

We illustrate this issue by presenting an example using the face dataset (Tenenbaum et al., 2000). This dataset contains 698 images (of size 64×64 and being further condensed to 32×32). The pose of each image is defined by three variables in T: *Light*, *Pan* and *Tilt*, as shown in Figure 2.1 (a). We adopt GLLiM to predict

these T's (low-dimensional) using the image (high-dimensional). The superiority of GLLiM in prediction, comparing to multiple existing approaches, for this data set was numerically illustrated in Deleforge et al. (2015).

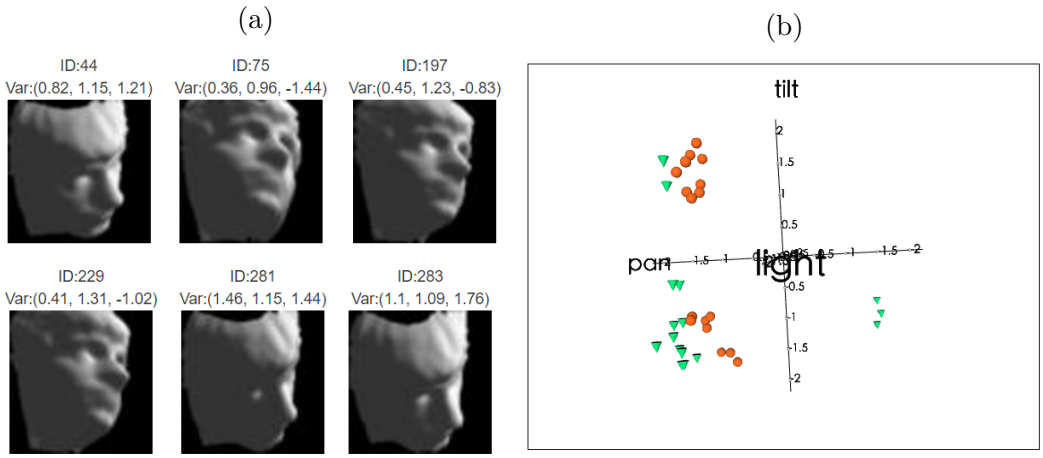


Figure 2.1: The clustering results of the face dataset obtained from GLLiM: (a) six face images from Cluster 7; (b) scatter plot of T for points within Cluster 7 and 13 clustered by GLLiM. Data points are from Cluster 7 (circle) and Cluster 13 (triangle). The three variables are (*Light*, *Pan*, *Tilt*).

Figure 2.1(b) shows the scatter plot of T within Clusters 7 and 13, grouped by GLLiM. By visual inspection, both clusters seem to consist of two or more sub-clusters. In GLLiM, samples within the same cluster are assumed to follow Gaussian distributions. This sub-cluster structure, however, violates the assumption and potentially increases the prediction errors. We demonstrate the difference in prediction performance before and after accounting for the sub-cluster structure in Table 2.1. We use prediction squared error (SE) for testing data pre- and post cluster-division. We observe that the prediction errors are mostly reduced if we account for the sub-cluster structure.

Dividing samples at low dimensions is an effective and straightforward solution for this issue. However, we could obtain small sub-clusters after division and then increase the prediction variance. In Table 2.1, Images 114 and 223 were assigned to small and/or tight local clusters and the prediction of T for these two images became

Image ID	GLLiM cluster	Original SE	Post-Division SE	Improved ratio
56	7	0.306	0.043	86.03%
223	7	0.016	0.180	-1039.83%
293	7	0.060	0.023	61.27%
302	7	0.087	0.003	96.99%
114	13	0.114	0.118	-2.93%
204	13	0.307	0.073	76.19%
294	13	3.119	0.120	96.15%

Table 2.1: The comparison of original and post cluster-division SE. The improved ratio is calculated as the ratio of difference of SE from pre- to post cluster-division over the original SE. The value is positive if the procedure reduces the SE, and negative otherwise.

worse after cluster-division. Conceptually, small clusters could damage prediction performance for several reasons: the small number of observations in such a cluster leads to estimates with large variation; a small cluster with a small covariance matrix determinant (volume) could lead to instability of the whole likelihood-based algorithm, and a small/tight cluster could consider a close-by testing sample unfit and force it to be predicted by another less suitable cluster with a larger within-cluster covariance. The last consideration is not relevant to model building but plays an important role in prediction precision.

This observation motivates us to look into enhancing prediction stability by controlling cluster sizes and eliminating outliers in the training dataset.

2.2 Hierarchical Gaussian Locally Linear Mapping (HGLLiM)

Hierarchical Gaussian Locally Linear Mapping (HGLLiM) is proposed to strike a balance between model flexibility and variation reduction in the estimated predictive model, with the goal of predicting the low-dimensional observable variables, T, using the high-dimensional X. This predictive model does not need to be the true model but should be effective in prediction. To present the fundamental concepts with clarity, we will first describe the model structure when $Y = T$, with minimum required notations.

The scenario of Y containing W is easily extended in Section 2.2.2.

2.2.1 Model description

The joint probability, $p(X = x, Y = y; \theta)$, of high-dimensional predictor X and low-dimensional response Y can be written as:

$$\sum_{k=1}^{K}\sum_{l=1}^{M} p(X = x|Y = y, Z = k, U = l; \theta)p(Y = y|Z = k, U = l; \theta)p(Z = k, U = l; \theta),$$

where θ denotes the vector of parameters; Z and U are, respectively, latent global and local cluster assignments. The locally linear relationship between X and Y is given by the mixture model below:

$$X = \sum_{k=1}^{K}\sum_{l=1}^{M} \mathbb{I}(Z = k, U = l)(A_{kl}Y + b_{kl} + E_k),$$

where $\mathbb{I}$ is the indicator function, $A_{kl} \in \mathbb{R}^{D\times L}$ and $b_{kl} \in \mathbb{R}^{D}$ map Y onto X, and $E_k \in \mathbb{R}^{D\times D}$ is the error term that absorbs the remaining uncertainty. Recall that D and L are dimensions of X and Y, respectively, and $D >> L$. Here, we let the local cluster size $M(k) \equiv M$ for notational simplicity only. We assume, within the k-th global cluster, that all local clusters share the same error structure, which follows a zero-mean Gaussian distribution with covariance matrix Σ_k. That is, we have,

$$p(X = x|Y = y, Z = k, U = l; \theta) = \mathcal{N}(x; A_{kl}y + b_{kl}, \Sigma_k).$$

The model is completed by assuming Y follows a mixture of Gaussian and defining a prior for clustering assignment:

$$p(Y = y|Z = k, U = l, \theta) = \mathcal{N}(y; c_{kl}, \Gamma_{kl}),$$
$$p(Z = k, U = l, \theta) = \rho_{kl},$$

where $c_{kl} \in \mathbb{R}^L$, $\Gamma_{kl} \in \mathbb{R}^{L \times L}$ and $\sum_{k=1}^{K} \sum_{l=1}^{M} \rho_{kl} = 1$. The vector of parameters in the inverse regression model, θ, is given by

$$\theta = \{c_{kl}, \Gamma_{kl}, \rho_{kl}, A_{kl}, b_{kl}, \Sigma_k\}_{k=1,l=1}^{K,M}.$$

The inverse conditional density can be written as:

$$p(X = x|Y = y; \theta) = \sum_{k=1}^{K} \sum_{l=1}^{M} \frac{\rho_{kl} \mathcal{N}(y; c_{kl}, \Gamma_{kl})}{\sum_{i=1}^{K} \sum_{j=1}^{M} \rho_{ij} \mathcal{N}(y; c_{ij}, \Gamma_{ij})} \mathcal{N}(x; A_{kl}x + b_{kl}, \Sigma_k),$$

and the conditional density in the forward regression model is expressed as:

$$p(Y = y|X = x; \theta^*) = \sum_{k=1}^{K} \sum_{l=1}^{M} \frac{\rho^*_{kl} \mathcal{N}(x; c^*_{kl}, \Gamma^*_{kl})}{\sum_{i=1}^{K} \sum_{j=1}^{M} \rho^*_{ij} \mathcal{N}(x; c^*_{ij}, \Gamma^*_{ij})} \mathcal{N}(y; A^*_{kl}y + b^*_{kl}, \Sigma^*_{kl}),$$

where θ^* denotes the parameter vector in the forward regression model:

$$\theta^* = \{c^*_{kl}, \Gamma^*_{kl}, \rho^*_{kl}, A^*_{kl}, b^*_{kl}, \Sigma^*_{kl}\}_{k=1,l=1}^{K,M}.$$

Note that θ^* has closed-form expressions as functions of θ, which makes it computationally efficient. The relation is obtained analytically with:

$$\begin{aligned}
c^*_{kl} &= A_{kl} c_{kl} + b_{kl}, \\
\Gamma^*_{kl} &= \Sigma_k + A_{kl} \Gamma_{kl} A_{kl}^\top, \\
\rho^*_{kl} &= \rho_{kl}, \\
A^*_{kl} &= \Sigma^*_{kl} A_{kl}^\top \Sigma_k^{-1}, \\
b^*_{kl} &= \Sigma^*_{kl} (\Gamma_{kl}^{-1} c_{kl} - A_{kl}^\top \Sigma_k^{-1} b_{kl}), \\
\text{and } \Sigma^*_{kl} &= (\Gamma_{kl}^{-1} + A_{kl}^\top \Sigma_k^{-1} A_{kl})^{-1}.
\end{aligned}$$

The prediction can be done by taking the expectation over the forward conditional density:

$$\mathbb{E}[Y|X=x] = \sum_{k=1}^{K}\sum_{l=1}^{M} \frac{\rho_{kl}^{*}\mathcal{N}(x; c_{kl}^{*}, \Gamma_{kl}^{*})}{\sum_{i=1}^{K}\sum_{j=1}^{M}\rho_{ij}^{*}\mathcal{N}(x; c_{ij}^{*}, \Gamma_{ij}^{*})}(A_{kl}^{*}x + b_{kl}^{*}) \tag{2.12}$$

2.2.2 HGLLiM model with partially-latent responses

Recall that the low-dimensional data $Y \in \mathbb{R}^{L}$ contains a latent component W. Namely, $Y^{\top} = (T^{\top}, W^{\top})$, where $T \in \mathbb{R}^{L_t}$ is observed and $W \in \mathbb{R}^{L_w}$ is latent and thus $L = L_t + L_w$. It is assumed that T and W are independent given Z, and so are W and U. According to the decomposition of Y, the corresponding mean (c_{kl}), variance (Γ_{kl}) and regression parameters (A_{kl}) of Y, at the local-cluster level, are given as:

$$c_{kl} = \begin{bmatrix} c_{kl}^{t} \\ c_{k}^{w} \end{bmatrix}, \ \Gamma_{kl} = \begin{bmatrix} \Gamma_{kl}^{t} & 0 \\ 0 & \Gamma_{k}^{w} \end{bmatrix}, \ \text{ and } A_{kl} = [A_{kl}^{t} \ A_{k}^{w}]. \tag{2.13}$$

That is, when $Z = k$, $U = l$, at the local-cluster level, $T \sim \mathcal{N}(c_{kl}^{t}, \Gamma_{kl}^{t})$; when $Z = k$, at the global-cluster level, $W \sim \mathcal{N}(c_{k}^{w}, \Gamma_{k}^{w})$. It follows that locally, the association function between the high-dimensional Y and low-dimensional X can be written as:

$$X = \sum_{k=1}^{K}\mathbb{I}(Z=k)\left\{\sum_{l=1}^{M}\mathbb{I}(U=l)(A_{kl}^{t}T + b_{kl}) + A_{k}^{w}W + E_{k}\right\}. \tag{2.14}$$

Finally, the parameter vector θ in the inverse regression model is rewritten as: $\theta = \{\rho_{kl}, c_{kl}^{t}, \Gamma_{kl}^{t}, A_{kl}^{t}, b_{kl}, c_{k}^{w}, \Gamma_{k}^{w}, A_{k}^{w}, \Sigma_{k}\}_{k=1,l=1}^{K,M}$.

It follows that (2.14) can be equivalently rewritten as

$$X = \sum_{k=1}^{K}\mathbb{I}(Z=k)\left\{\sum_{l=1}^{M}\mathbb{I}(U=l)(A_{kl}^{t}T + b_{kl}) + A_{k}^{w}c_{k}^{w} + E_{k}'\right\}, \tag{2.15}$$

where the error vector E'_k is modeled by a zero-centered Gaussian variable with a $D \times D$ covariance matrix given by

$$\Sigma'_k = \Sigma_k + A_k^w \Gamma_k^w A_k^{w\top}. \tag{2.16}$$

Considering realizations of variables T and X, the addition of the latent W naturally leads to a covariance structure, namely (2.16), where $A_k^w \Gamma_k^w A_k^{w\top}$ is at most of rank L_w. When Σ_k is diagonal, this structure is that of factor analysis with at most L_w factors, and represents a flexible compromise between a full covariance with $O(D^2)$ parameters on one side, and a diagonal covariance with $O(D)$ parameters on the other.

Using the same number of total clusters and considering the fact that Σ_k and A_k^w are only estimated at the global-cluster level, we note that the total number of parameters needed to model the covariances, Σ_k, and the latent transformation coefficients, A_k^w, using HGLLiM is $1/M$ of that required by using GLLiM. In addition, the key emphasis of HGLLiM is to conduct prediction. As shown in (2.13), (2.15), and (2.16) at the local vs. global-cluster levels, we now separate the estimation of the mean association functions, which play a key role in prediction, from that of high-dimensional covariance matrices, so that the means can be obtained even more locally. Together with the current dependence structures being stably estimated at the global-cluster level using more data points per cluster, the HGLLiM provides a strong prediction tool built on a structure facilitating sensible approximations to the true underlying distribution of low-dimensional T and high-dimensional X.

2.3 Estimation procedure

In this section, an EM algorithm for HGLLiM is provided for parameter estimation. We then present an extended version, tailored for ensuring stability and outlier

trimming. The selection of tuning parameters is also discussed.

2.3.1 The EM algorithm for HGLLiM

The HGLLiM model contains three sets of latent variables: $Z_{1:N} = \{Z_N\}_{n=1}^N$, $U_{1:N} = \{U_n\}_{n=1}^N$ and $W_{1:N} = \{W_n\}_{n=1}^N$. The first two sets of variables indicate the global and the local cluster assignments and the last one is the latent covariates. The model parameters, θ, as defined in Equation (2.12), can be estimated using the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). We can divide the EM algorithm for HGLLiM into several steps: an E-Z, U step for estimating the posterior probability of being assigned to a global or a local cluster, an E-W step for finding an estimation of latent variable W and a maximization step for estimating parameters at the local and global cluster levels.

E-Z, U Step:

We denote the posterior probability of observation n being assigned to global cluster k, local cluster l, based on the observed data, to be

$$r_{nkl} \;=\; p(Z_n = k, U_n = l | t_n, x_n; \theta); \tag{2.17}$$

and we let,

$$r_{nk} \;=\; p(Z_n = k | t_n, x_n; \theta). \tag{2.18}$$

The posterior probability of sample n being assigned to local cluster (k, l) is given by

$$\begin{aligned} r_{nkl} &= p(Z_n = k, U_n = l | t_n, x_n; \theta) \\ &= \frac{\rho_{kl} p(t_n, x_n | Z_n = k, U_n = l; \theta)}{\sum_{i=1}^K \sum_{j=1}^M \rho_{ij} p(t_n, x_n | Z_n = i, U_n = j; \theta)}, \end{aligned}$$

where $p(t_n, x_n | Z_n = k, U_n = l; \theta) = p(x_n | t_n, Z_n = k, U_n = l) p(t_n | Z_n = k, U_n = l)$. The first term is given by $p(x_n | t_n, Z_n = k, U_n = l) = \mathcal{N}(x_n; A_{kl}^t t_n + b_{kl} + A_k^w c_k^w, A_k^w \Gamma_k^w A_k^{w\top} +$

Σ_k). Recall that the second term $p(t_n|Z_n = k, U_n = l) = \mathcal{N}(t; c^t_{kl}, \Gamma^t_{kl})$.

A direct derivation shows that

$$\begin{aligned} r_{nk} &= p(Z_n = k|t_n, y_n; \theta) \\ &= \sum_{l=1}^{M} r_{nkl}. \end{aligned}$$

E-W Step:

The distribution $p(w_n|Z_n = k, t_n, x_n; \theta)$ can be shown to be Gaussian with mean μ^w_{nk} and covariance matrix S^w_k. The estimation of the mean and covariance matrix is given by:

$$\begin{aligned} \tilde{\mu}^w_{nk} &= \sum_{l=1}^{M} \frac{r_{nkl}}{r_{nk}} \tilde{S}^w_k \left(A^{w\top}_k \Sigma^{-1}_k (x_n - A^t_{kl} t_n - b_{kl}) + (\Gamma^w_k)^{-1} c^w_k \right), \\ \tilde{S}^w_k &= \left\{ (\Gamma^w_k)^{-1} + A^{w\top}_k \Sigma^{-1}_k A^w_k \right\}^{-1}. \end{aligned} \tag{2.19}$$

The maximization step consists of two sub-steps. The first step aims to estimate parameters for a Gaussian Mixture Model and the second one focuses on estimating parameters for mapping.

M-GMM Step:

In this step we only consider the parameters related to the Gaussian Mixture Model. In particular, we want to estimate $\{\rho_{kl}, c^t_{kl}, \Gamma^t_{kl}, \}^{K,M}_{k=1,l=1}$. Hereinafter, we let $r_{kl} = \sum_{n=1}^{N} r_{nkl}$ and $r_k = \sum_{n=1}^{N} r_{nk}$. We obtain:

$$\begin{aligned} \tilde{\rho}_{kl} &= \frac{r_{kl}}{N}, \\ \tilde{c}^t_{kl} &= \frac{\sum_{n=1}^{N} r_{nkl} t_n}{r_{kl}}, \\ \text{and } \tilde{\Gamma}^t_{kl} &= \frac{\sum_{n=1}^{N} r_{nkl} (t_n - \tilde{c}_{kl})(t_n - \tilde{c}^t_{kl})^\top}{r_{kl}}. \end{aligned}$$

M-mapping Step:

The M-mapping step aims to estimate $\{A_{kl}^t, b_{kl}, A_k^w, \Sigma_k\}_{k=1,l=1}^{K,M}$. It is assumed that T and W are independent given the cluster assignment. Based on this, we could update A_k^w first:

$$\tilde{A}_k^w = \tilde{X}_k \tilde{V}_k^\top (S_k^w + \tilde{V}_k \tilde{V}_k^\top)^{-1} \tag{2.20}$$

where

$$\begin{aligned}
\tilde{V}_k &= \frac{1}{\sqrt{r_k}}[\sqrt{r_{1k}}(\tilde{\mu}_{1k}^w - \tilde{\mu}_k^w), ..., \sqrt{r_{Nk}}(\tilde{\mu}_{Nk}^w - \tilde{\mu}_k^w)], \\
\tilde{X}_k &= \frac{1}{\sqrt{r_k}}[\sqrt{r_{1k}}(x_1 - \sum_{l=1}^{M} \frac{r_{1kl}}{r_{1k}} \tilde{x}_{kl}), ..., \sqrt{r_{Nk}}(x_N - \sum_{l=1}^{M} \frac{r_{Nkl}}{r_{Nk}} \tilde{x}_{kl})], \\
\tilde{\mu}_k^w &= \sum_{n=1}^{N} \frac{r_{nk}}{r_k} \tilde{\mu}_{nk}^w, \\
\tilde{x}_{kl} &= \sum_{n=1}^{N} \frac{r_{nkl}}{r_{kl}} x_n.
\end{aligned}$$

Note the difference between how X and V are being centered. For X, we center it against the local cluster mean, while we let V be centered at the global-cluster level. Once we obtain A_k^w we subtract the latent variables component from X and update A_{kl}^t and b_{kl}, accordingly. Letting $x_{nk}^* = x_n - \tilde{A}_k^w \tilde{\mu}_{nk}^w$, we get:

$$\begin{aligned}
\tilde{A}_{kl}^t &= \tilde{X}_{kl}^* \tilde{T}_{kl}^\top (\tilde{T}_{kl} \tilde{T}_{kl}^\top)^{-1}, \\
\tilde{b}_{kl} &= \sum_{n=1}^{N} \frac{r_{nkl}}{r_{kl}} (x_{nk}^* - \tilde{A}_{kl}^t t_n),
\end{aligned}$$

where

$$\tilde{T}_{kl} = \frac{1}{\sqrt{r_{kl}}}[\sqrt{r_{1kl}}(t_1 - \tilde{t}_{kl}), ..., \sqrt{r_{Nkl}}(t_N - \tilde{t}_{kl})],$$
$$\tilde{X}^*_{kl} = \frac{1}{\sqrt{r_{kl}}}[\sqrt{r_{1kl}}(x^*_{1k} - \tilde{x}_{kl}), ..., \sqrt{r_{Nkl}}(x^*_{Nk} - \tilde{x}_{kl})],$$
$$\tilde{t}_{kl} = \sum_{n=1}^{N} \frac{r_{nkl}}{r_{kl}} t_n,$$
$$\tilde{x}_{kl} = \sum_{n=1}^{N} \frac{r_{nkl}}{r_{kl}} x^*_{nk}.$$

Finally, we can update Σ_k by:

$$\tilde{\Sigma}_k = \tilde{A}^w_k \tilde{S}^w_k \tilde{A}^w_k + \sum_{n=1}^{N} \frac{r_{nk}}{r_k}[x_n - R_n][x_n - R_n]^\top, \tag{2.21}$$

where $R_n = \sum_{l=1}^{M} \frac{r_{nkl}}{r_{nk}}(\tilde{A}^t_{kl} t_n + \tilde{b}_{kl}) - \tilde{A}^w_k \tilde{\mu}^w_{nk}$.

2.3.2 Robust estimation procedure

The EM algorithm as stated in Section 2.3.1 gives the key structure of the method. However, even with the inversion step, the prediction procedure still involves a high-dimensional predictor and elevated variation in estimated parameters, induced by small clusters or abnormal observations, that could lead to reduced prediction quality. Stability can be achieved by constraining the sizes of the clusters (controlling both covariance volume and prediction variance) and trimming outliers. We design a robust estimation procedure to refine the standard EM algorithm with the purpose of enhancing model stability, which consequently leads to improved prediction performance.

Let $\sum_{n=1}^{N} r_{nkl}$ represent the cluster size for cluster (k, l). Each data point in a cluster whose cluster size is smaller than a pre-determined *minSize* is reassigned to other clusters. The point is kept when the prediction squared error is less than a

pre-determined *dropThreshold*; otherwise, it would be excluded from the current EM iteration when updating the estimated parameters. With the data points within a cluster playing a dominating role in estimating within-cluster parameters, the cluster size plays the role of the sample size in estimation: when the sample size is too small, the prediction quality deteriorates even if the assumed structure is correct. Improved prediction performance might be achieved by assigning such a data point within a small cluster to another cluster that shares similar structures. If no such a cluster can be identified, then the data point is excluded from the construction of the prediction model. The algorithm is shown in Algorithm 2.1 and described as follows:

1. The algorithm is initialized by adopting the parameters θ, mean and covariance, $\tilde{\mu}_{nk}^{w}$, $\tilde{S}_{k}^{w}$, of latent W of the k-th cluster, and cluster assignment r_{nkl} obtained from the EM algorithm described in Section 2.3.1.

2. The estimating procedure iterates through the following sub-steps until the algorithm converges:

   (a) Trimming step: In order to remove outliers, we scan through all local clusters and remove all samples whose in-sample prediction squared errors are greater than a pre-determined *dropThreshold*. The prediction squared error for the n-th sample is calculated as:

   $$E_n^2 = ||t_n^{pred} - t_n||_2^2, \tag{2.22}$$

   where t_n is the true value and t_n^{pred} is the prediction from Equation (2.12). Note that the low-dimensional data $\{t_n\}_{n=1}^{N}$ are standardized before training so that each dimension would be equally weighted. The samples with in-sample prediction squared error larger than *dropThreshold* are considered outliers and are temporarily removed by assigning r_{n^*kl} to be 0 at that

iteration of the maximization step, where n^* indicates the training sample whose $E^2_{n^*} > dropThreshold$.

(b) Maximization step with a cluster size constraint: The estimation of θ is the same as in the Maximization step described in Section 2.3.1 but with an additional cluster size constraint. Before estimating parameters for each local cluster (k, l), we first check the associated cluster size. If the cluster size is smaller than the given $minSize$, we force the training data originally assigned to this cluster to either be assigned to other clusters during the E-step in updating cluster-assignment Z, and U, or, if no appropriate cluster can be found, to be trimmed during the next Trimming Step.

(c) Update step for the latent variables: Estimation of $\tilde{\mu}^w_{nk}$, $\tilde{S}^w_k$ and r_{nkl} are done using the E-W and E-Z, U step described in Section 2.3.1.

Algorithm 2.1 Robust estimation procedure for HGLLiM

Input : Observed data pair $\{t_n, x_n\}_{n=1}^N$; global cluster number K; local cluster number M; prediction trim threshold $dropThreshold$; minimum cluster weight $minSize$.

Output : Model parameter θ fulfills the requirement of the prediction error and the minimum cluster size.

while *the likelihood does not converge* **do**

`// Trim`

Calculate prediction t^{pred} for all n

Update $r_{nkl} \leftarrow 0$ for all n such that $||t_n^{pred} - t_n||_2^2 > dropThreshold$

`// M step with cluster weight constraint` $minSize$

Compute cluster weight $r_{kl} \leftarrow \sum_{n=1}^N r_{nkl}$

for $k = 1$ **to** K **do**

for $l = 1$ **to** M **do**

if $r_{kl} < minSize$ **then**

remove cluster (k, l)

else

$\theta_{kl} \leftarrow Maximization(t, x, r_{nkl}, \mu_k^w, S_k^w)$

end

end

end

`// E-step`

Update r_{nkl}, μ_k^w and S_k^w using θ

end

2.3.3 Tuning parameter selection

For HGLLiM, there are several user-defined parameters: the dimension of the latent variables L_w, the number of global clusters K, the number of local clusters

M, the minimum allowed cluster size $minSize$ and the maximum allowed in-sample prediction error $dropThreshold$. Through the changes in these tuning parameters, the algorithm can be used to analyze all kind of data. We identify default recommendations for certain parameters that work for almost all cases, and suggest simple procedures that can be used to select others.

- K and L_w: The number of clusters, K, reflects the number of local linear associations between covariates and responses. On the other hand, the number of latent factors, L_w, models the variation that cannot be captured by these linear associations. The combination of (K, L_w) influences the ability to capture the mean and covariance structure of the relationship between X and Y. Selecting K and L_w through cross-validation is time-consuming, particularly because there could be a large set of potential K to be considered. We propose a method to restrict the searching space via the use of BIC. Using the face dataset as an example, Table 2.2 shows the cluster number selected using BIC when L_w is fixed, while Table 2.3 shows the number of latent factors selected by BIC when K is fixed. These two tables show the roles played by K and L_w in terms of how they compensate for each other. The model complexity increases as we increase K or L_w. Therefore, BIC prefers the combination of either a small K with a large L_w or a large K with a small L_w. It is also known that BIC is conservative. Thus the parameters are most likely underestimated. Though it matters less here, with additional sub-clustering steps in HGLLiM, we slightly adjust the K and L_w selected by BIC to improve prediction performance. We construct a search grid of K and L_w as follows. First, we select K using BIC under a small L_w. This cluster number is called K^{BIC}. Next, we fix the cluster number to K^{BIC} and select the corresponding number of latent factors, $L_w^{K^{BIC}}$. To identify the possible range of K and L_w, we increase the cluster number and select the corresponding number of latent factors. As an example, we could set

	L_w=0	L_w=1	L_w=2	L_w=8	L_w=9	L_w=10
BIC	-8.75e+05	-9.35e+05	-9.48e+05	-1.08e+06	-1.09e+06	-1.11e+06
K	14	13	10	6	6	6

Table 2.2: The value of BIC and K selected by BIC for a given L_w. For a fixed L_w, row 1: the minimum value of BIC; and row 2: the number of clusters, K, that achieves this BIC.

	K=5	K=10	K=15	K=20	K=25	K=30	K=35	K=40
BIC	-1.11e+06	-1.03e+06	-9.72e+05	-9.33e+05	-8.90e+05	-8.53e+05	-8.14e+05	-8.09e+05
L_w	10	8	7	3	1	1	0	0

Table 2.3: The value of BIC and the L_w selected by BIC for a given K. For a fixed K, row 1: the minimum value of BIC; and row 2: the dimension of W, L_w, that achieves this BIC.

the cluster number as $K^{BIC} + 15$ and find the corresponding number of latent factors, $L_w^{K^{BIC}+15}$, again by BIC. Note that $L_w^{K^{BIC}+15}$ is smaller than $L_w^{K^{BIC}}$. If not, we can use $K = K^{BIC} + 20$ or even $K^{BIC} + 25$, until the resulting L_w is smaller than $L_w^{K^{BIC}}$, and this K would be the upper bound we use for values of K. Applying an equal consideration of preventing being too conservative, we could extend the search range of $L_w^{K^{BIC}}$ to $L_w^{K^{BIC}} + 2$. Finally, cross-validation is conducted within the range of $(K^{BIC}, K^{BIC} + 15)$ and $(L_w^{K^{BIC}+15}, L_w^{K^{BIC}} + 2)$ for searching for the combination of K and L_w that achieves the best performance.

- M: It is assumed that there would be one or more local clusters within each global cluster. The choice of M depends on the nature of the data structure. We found that the final result would not be sensitive to M; the EM algorithm combined with the refining algorithm would adjust itself and unneeded local clusters would be dissolved.

- $minSize$: A two-dimensional grid search cross-validation algorithm can be used to search for the best combination of $minSize$ and $dropThreshold$, and we

have explored this option. However, this practice could be time-consuming. To obtain an appropriate suggested value for $minSize$ we calculate the matrix volume of Γ^*_{kl}, the covariance matrix used in prediction, and look for the drop-off. Using the face dataset as an example, we implement HGLLiM with $K = 15$, $M = 5$ and set $L_w = 2$. The volume of Γ^*_{kl} is approximated by the product of the top three eigenvalues. Figure 2.2 shows the relationship between volumes of Γ^*_{kl} versus cluster sizes. A small covariance matrix is likely to cause a surge in likelihood and difficulties for the nearby testing sample to be classified as a member of the cluster, both leading to inflation of the prediction mean squared error. Figure 2.2 suggests that small covariance matrices could be expected when the cluster size is smaller than 4. In view of this, we set $minSize = 5$ for this case. Our empirical experiences imply that this simple approach leads to outcomes comparable to those of the more complicated two-dimensional grid search algorithm.

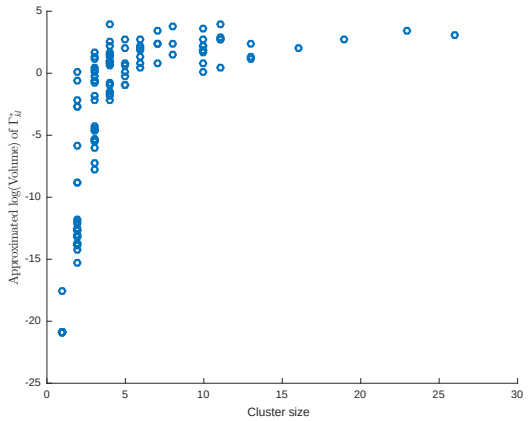


Figure 2.2: The logarithm of the approximated volume of Γ^*_{kl} against the cluster size.

- $dropThreshold$: With $minSize$ being fixed, $dropThreshold$ could be simply estimated by a K-fold cross-validation. From the experimental results, we establish that the prediction mean squared error is not sensitive to the choice of $dropThreshold$ within a reasonable range. We show this using the outcomes in Section 2.4.

2.4 Numerical results

In this section, we analyze two moderate-size datasets to demonstrate how the method works for data with different features and the insensitivity of tuning parameter selection on a wide range of selection domains. The analysis of two large-scale complex datasets can be seen in Chapter III. Key features of each dataset, and thus the type of data they represent, are reported in the corresponding subsections. Throughout, we use squared error (Equation (2.22)) to evaluate the prediction performance for each data point. We also calculate the prediction mean squared error (MSE) among all testing samples with $MSE = \sum_{n=1}^{N_{test}} E_n^2/N_{test}$, where N_{test} is the total number of testing samples.

We calculate and compare the MSE or the quantiles of squared errors across several methods:

1. HGLLiM: This is the proposed method. The user-defined parameters K and L_w are set to values using the method described in Section 2.3.3. The number of local clusters M is set to 5 to reflect the possible sub-cluster structure. In each global cluster, the number of local clusters varies and depends on the structure of the dataset. Some of the local clusters would be dissolved so the number of local clusters could be less than M. The initial cluster assignment is done by dividing the GLLiM clustering outcomes at the low dimension using the R package *mclust* (R Core Team, 2019; Scrucca et al., 2017). As stated before, the robust version of the EM algorithm is used throughout the experiments. We set $minSize = 5$ for all of the analyses and post-analysis checks in the neighborhood of 5 suggest this is an appropriate choice. The prediction MSE using different values of $dropThreshold$ would be calculated and compared.

2. GLLiM: The original GLLiM algorithm. GLLiM is compared to other methods under the same settings of K and L_w. The initial cluster assignment is done

by applying a Gaussian mixture model to a dataset that combines the low-dimensional T and high-dimensional X together.

3. GLLiM-structure: This method adopts the number of clusters learned structurally by HGLLiM. In addition, outliers identified by HGLLiM are removed from the training dataset. We adopt the same tuning parameters as GLLiM and the initial conditions are obtained from the outcomes of HGLLiM. The key difference between GLLiM-structure and HGLLiM is that GLLiM-structure uses locally estimated covariance, which may be more appropriate for a large dataset with more local dependence features. Its effectiveness also suggests an additional usage of HGLLiM, in terms of structure learning and identification of outliers.

2.4.1 The face dataset

The face dataset, consisting of 698 samples, was analyzed in the original GLLiM paper (Deleforge et al., 2015). For this dataset, we are interested in predicting the pose parameters ($L_t = 3$) using the image information. The size of each image is condensed to 32×32, and thus $D = 1024$. In addition, T is standardized so that all three dimensions are equally weighted. The histograms of the three T variables bear no clustering structure. Consequently, the mixture modeling serves the purpose of local linear approximation and inverse regression is utilized to circumvent the difficulties encountered in high-dimensional regression.

In each run of cross-validation investigation, we follow the procedure in Deleforge et al. (2015) and select 100 testing samples and keep the remaining 598 as training samples. We repeat this procedure 20 times to establish 2000 testing samples. According to the approach described in Section 2.3.3, we run cross-validation on K from 10 to 25, L_w from 1 to 15. The cross-validation results in Figure 2.3(a) suggest that $K = 20$, $L_w = 9$. It is noted that the prediction errors decrease with increasing

values of L_w. This phenomenon suggests that the high-dimensional X are dependent and that accounting for such dependency via the latent W leads to improvement in prediction. It is also observed that the change in prediction error is relatively small when L_w exceeds a certain value. Therefore, we fix the number of latent factors and compare the prediction performance under $K = 10$, $K = 15$ and $K = 25$.

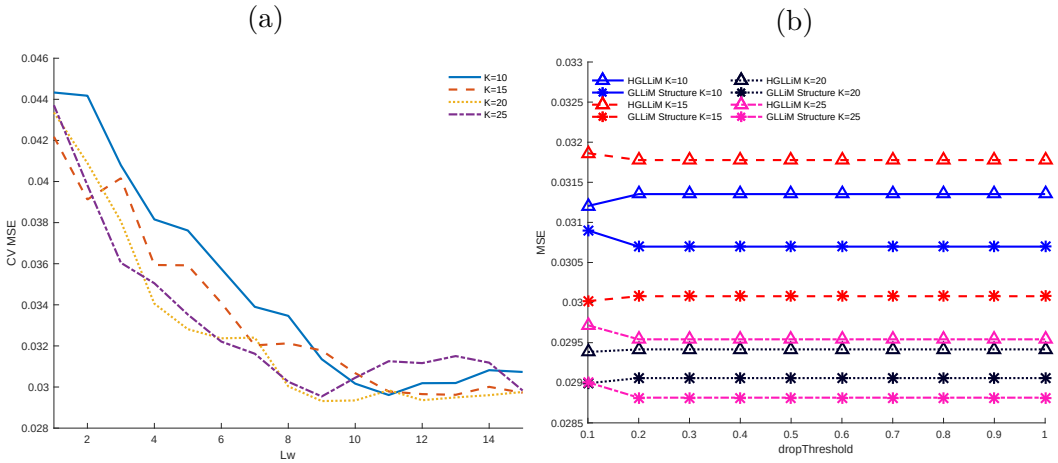


Figure 2.3: Results for different user-defined parameters of the face dataset. (a) The HGLLiM cross-validation results for different K and L_w. (b) The prediction MSE of different K and different methods against different *dropThresholds*.

Figure 2.3(b) shows prediction outcomes under different values of *dropThreshold*. We observe that for different methods and different K, the prediction MSEs are not sensitive to the values of *dropThreshold*. Thus, we compare the prediction MSE of HGLLiM and GLLiM-structure when *dropThreshold* $= 0.5$ to GLLiM in Table 2.4. The prediction MSE for GLLiM decreases as K increases, which indicates that more clusters could be helpful to capture the non-linear relationship between X and T. For HGLLiM, we observe that the prediction MSE is not sensitive to the choice of K. In addition, the numbers of clusters are similar under different choices of K. This indicates that HGLLiM could adjust itself to reach the number of clusters suitable to its setting. As for GLLiM-structure, the prediction MSEs are slightly smaller than those of HGLLiM. This is because GLLiM-structure estimates all parameters using local clusters and local covariances, and the prediction would be less biased

	K=10		K=15		K=20		K=25	
	MSE	#Cluster	MSE	#Cluster	MSE	#Cluster	MSE	#Cluster
GLLiM	0.0711	10.00	0.0441	15.00	0.0369	20.00	0.0321	25.00
HGLLiM	0.0314	43.90	0.0318	51.35	0.0294	53.75	0.0295	53.45
GLLiM-structure	0.0307	43.90	0.0301	51.35	0.0291	53.75	0.0288	53.45

Table 2.4: The prediction MSE and the average cluster number of the face dataset when $dropThreshold = 0.5$.

when the local structures sufficiently differ. In the face dataset, there is no obvious cluster structure and, as a result, clustering only serves the purpose of improving local approximation. Thus, the prediction MSE for GLLiM-structure would be smaller. However, the differences in prediction MSEs between HGLLiM and GLLiM-structure are small, which implies that the settings learned from HGLLiM are appropriate, even though HGLLiM imposes a global-cluster structure when there is none. Overall, the prediction performance for HGLLiM is similar when $K = 20$ and $K = 25$. As for GLLiM-structure, the MSE is smaller when $K = 25$ but the difference is negligible.

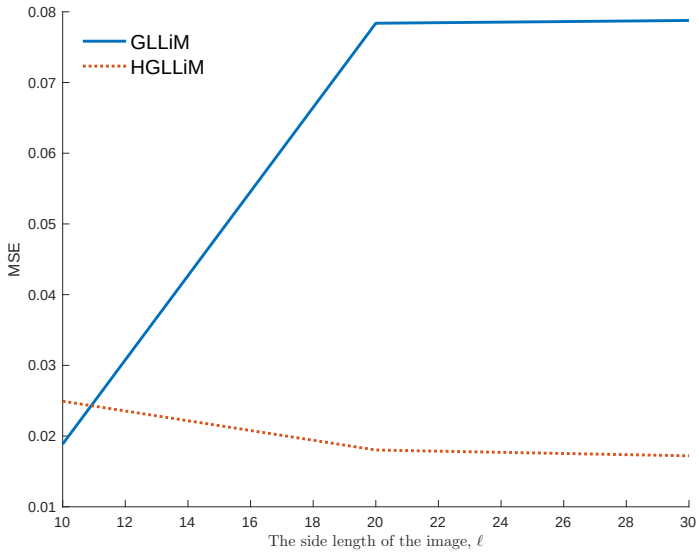


Figure 2.4: The prediction MSE of the face dataset under different dimensions of X. Each image in the face dataset consists of $\ell \times \ell$ pixels, where ℓ, the side length of the image, is the square root of the dimension of X.

We further investigate the phenomenon described in Section 2.1. Specifically, as the dimension of X becomes higher, not only does the number of covariance param-

eters increase, but there is also a higher chance the clusters formed by GLLiM could contain sub-clusters and/or outliers, which could decrease the prediction quality. We use Cluster 7 as our reference to create two clusters. There are two sub-clusters within Cluster 7. We first identify the center of each sub-cluster using the low-dimensional T and find the 30 nearest samples to each center. We randomly select 25 data points from each sub-cluster as the training data and use the rest of the data points as the testing samples. Thus, there will be 50 training samples and 10 testing samples. The procedure is repeated 20 times and the results are aggregated together to evaluate the model performance.

To investigate the prediction performance under the different dimensions of X, we resize the face image to $\ell \times \ell$ pixels, where we denote ℓ the side length of the image so that the dimension of X is $D = \ell \times \ell$. For GLLiM, we set the number of clusters, K, to be 2 and the dimension of the latent variables, L_w, to be 9. For HGLLiM, we have one global cluster and two local clusters, that is, $K = 1, M = 2$. As suggested in Figure 2.3(a), we let $L_w = 9$ since this setting results in smaller cross-validation MSE. We disable the robust estimation step, which is equivalent to setting $minSize = 0$, $dropThreshold = \infty$, as described in Section 2.3.2; also see Section 2.3.3. Figure 2.4 shows the result of prediction MSE under different dimensions of X. When the dimension of X is low, GLLiM can outperform HGLLiM. However, as the dimension of X increases, we observe that the prediction error of GLLiM increases, suffering from the potentially less suitable cluster assignments. On the other hand, HGLLiM maintains appropriate clustering results, and thus the prediction performance remains similar for all image sizes, if not slightly improved with the increasing dimension of X and the enhanced information in the images with higher resolution.

2.4.2 The orange juice dataset

The orange juice dataset contains the spectra measured on different kinds of orange juice ($N = 218$). The goal is to use the spectra to predict the level of sucrose ($L_t = 1$). We follow the step described in Perthame et al. (2018) and decompose the spectra on a spline basis with ($D = 134$) to make $D \approx N$. This dataset is known for the presence of outliers; the realization of X and T is given in Figure 2.5.

We set up the following prediction evaluation procedure. In each run, we randomly select 20 testing samples from the main population (excluding outliers). The remaining 198 samples (including outliers, unless otherwise specified) are used for training. These outliers were identified through Leave One Out Cross Validation (LOOCV) using GLLiM, with $K = 10$ and $L_w = 2$. Although the set of outliers may differ for different selections of K, L_w, the severe outliers are always selected and they are included here. We identify 11 points, which are the observations with the top 5% of the prediction E^2's (above 4.8) among all data points, as outliers. Removing outliers from testing data prevents the summarized outcomes from being overwhelmed by the prediction results of few points, which potentially makes the differences among methods less obvious. All methods were evaluated using the same settings.

Figure 2.6(a) shows the cross-validation results, which suggest the use of $K = 5$, $L_w = 8$. For comparison purposes, we also provide MSE results for $K = 10$ and $K = 15$. The rest of the setting is the same as the experimental setting used for the face dataset.

To evaluate the influence of outliers on GLLiM, we conduct an analysis in which we use the same cluster number as in GLLiM-structure but without removing training outliers. This method is referred to as GLLiM-outlier. In addition, we consider SLLiM in Perthame et al. (2018), provided by the R package *xLLiM* (Perthame et al., 2017). SLLiM is a counterpart of GLLiM that accommodates abnormal samples using Student's t-distributions. Precisely, the high-dimensional X is modeled by a

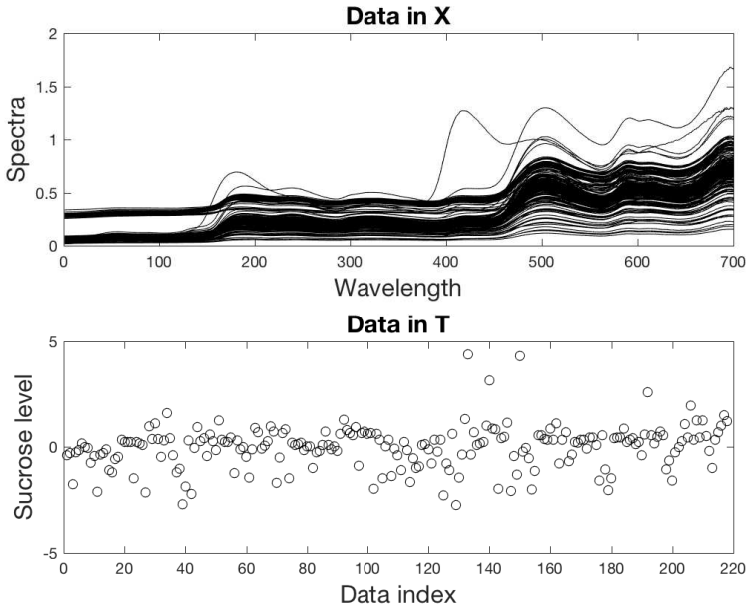


Figure 2.5: The orange juice dataset. The upper panel shows the high-dimensional data (X) and the lower one shows the low-dimensional data (T).

mixture of K generalized multivariate Student's t-distributions, using the structure given in Section 5.5 (p.94) of Kotz and Nadarajah (2004). We also compare SLLiM performances by using the same cluster number learned structurally by HGLLiM. We refer to the resulting procedure as "SLLiM-structure." We use the default settings in *xLLiM* for the remaining SLLiM configurations.

Figure 2.6(b) shows the prediction MSE for different *dropThresholds*. The prediction MSEs vary, mainly reflecting the high variation in this dataset, partially due

	K=5		K=10		K=15	
	MSE	#Cluster	MSE	#Cluster	MSE	#Cluster
GLLiM	0.1259	5.00	0.1210	10.00	0.0918	15.00
HGLLiM	0.0587	9.95	0.0681	11.85	0.0692	12.80
GLLiM-structure	0.0621	9.95	0.0742	11.85	0.0746	12.80
GLLiM-outlier	0.0976	9.95	0.1171	11.85	0.1044	12.80
SLLiM	0.1039	5.00	0.0788	10.00	0.0706	15.00
SLLiM-structure	0.0907	9.95	0.0747	11.85	0.0721	12.80

Table 2.5: The prediction MSE and the average number of clusters of the orange juice dataset when $dropThreshold = 0.5$.

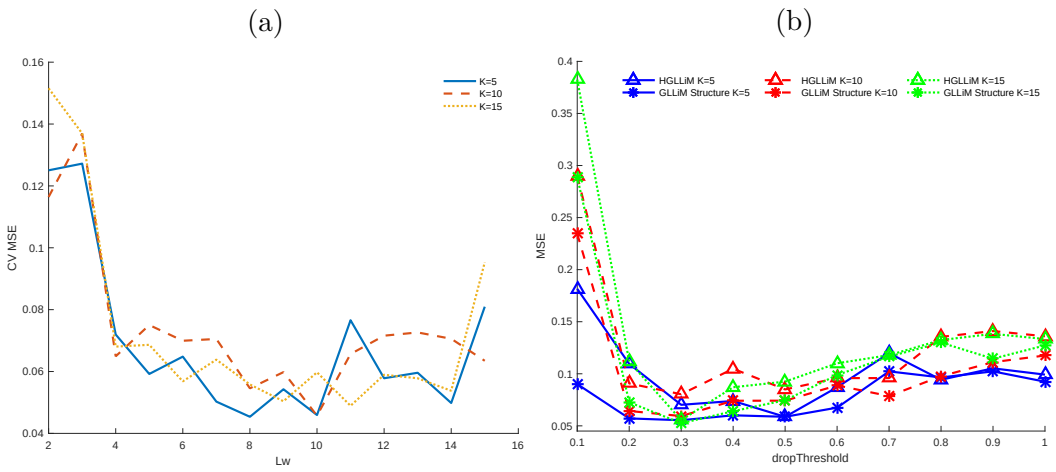


Figure 2.6: Results for the user-defined parameters of the orange juice dataset. (a) The HGLLiM cross-validation results for different K and L_w. (b) The prediction MSE of different K and different methods against different $dropThresholds$.

to outliers. For a small $dropThreshold$, the number of identified training outliers is higher than expected. This reduces the training data size and makes the prediction unreliable. As $dropThreshold$ reaches a reasonable value, the prediction performance becomes better. However, more and more abnormal training samples are included in the training dataset as $dropThreshold$ keeps increasing. These outlying data enlarge the model variance and downgrade the prediction performance. Table 2.5 shows the results for $dropThreshold = 0.5$. We observe that for $K = 5$, the cluster number is not sufficiently large for GLLiM to capture the non-linear trend in the data, which results in a relatively large prediction MSE. HGLLiM, on the other hand, adjusts the cluster number automatically and the prediction errors are smaller. In addition, HGLLiM removes training outliers that would decrease the model performance. This explains why even though the cluster number is as large as $K = 15$ (larger than the average size of 12.8 used in GLLiM-structure), GLLiM still suffers from large prediction errors. We further observe the benefit of removing outliers by comparing GLLiM-structure and GLLiM-outlier. The prediction errors for GLLiM-structure are smaller than those produced by GLLiM-outlier, and the only difference between GLLiM-structure and GLLiM-outlier is whether training outliers, identified by HGLLiM, are

removed. There are 11 outliers in the training dataset. HGLLiM could effectively identify and remove all of them. In addition to these outliers, some potential outlying samples that could result in an unstable model are trimmed as well. Overall, about 6% to 10% of the training samples would be removed by HGLLiM.

SLLiM and SLLiM-structure use t-distributions to accommodate the existence of outliers. They are expected to perform better than their Gaussian counterparts (GLLiM and GLLiM-outlier). When K, the cluster number, is small, there would be more samples in each cluster and thus the cluster size, $\sum_{n=1}^{N} r_{nkl}$ for cluster (k, l), would be large. In contrast, when K is large, samples would be divided into more clusters, which decreases the cluster size. It is observed that when K is small, accommodating outliers with t-distributions is not as effective as removing them by comparing SLLiM-structure and GLLiM-structure. When the number of clusters becomes larger, outliers can be assigned to a cluster with less influence on the prediction and thus we can obtain similar prediction performance from SLLiM-structure and GLLiM-structure. However, removing outliers would reduce the cluster size and result in unstable prediction performance. To provide reliable model performance, HGLLiM controls the cluster size via the tuning parameter $minSize$. In addition, HGLLiM estimates the covariance matrices under global-cluster level, and this estimation is more reliable compared to GLLiM-structure, which estimates covariance matrices locally. SLLiM does not remove any samples, and thus the performance would be better than that of GLLiM-structure when the cluster number, K, is large. Although removing outliers is more effective, accommodating outliers may still be an alternative to combat outliers when the cluster size is the concern.

2.5 Conclusion

We propose HGLLiM as a parsimonious and structured version of GLLiM. HGLLiM adopts a two-level hierarchical structure of clusters. The assumed structure enables

us to assess the parameters in the mean association functions more locally without suffering from the clustering outcomes being dominated by the dependence structures in the high-dimensional predictors. Under the same construction, we also estimate the reduced number of covariance parameters with more data points. In addition, we implement a robust version of HGLLiM to enhance model stability and reduce prediction variation. HGLLiM further leads to a post-learning version of GLLiM, called GLLiM-structure. By using local means and local variances, with unfitted points removed, GLLiM-structure tends to reach improved empirical performances.

The motivation behind HGLLiM and GLLiM-structure is to obtain precise predictions by constructing stable training models. Eliminating the existence of small clusters and removing outliers assist in achieving this goal. The fact that HGLLiM only focuses on preserving primary structures learned from the training dataset may reduce the quality of its predictions of rare data points, which are insufficiently presented therein. Nevertheless, by utilizing the largest membership posterior probability r_{nkl} among all clusters (k, l) and by recognizing when this maximum is likely to be much smaller than those obtained from the majority of the data, we can identify such testing samples with unreliable prediction results.

Despite the drawback that the resulting training model obtained by HGLLiM may or may not reflect the exact true model that generates all the data, it nevertheless captures the critical structure and establishes a model that can be stably estimated using the data available. HGLLiM can still provide reliable prediction outcomes for the majority of the data and is the recommended approach as a starting point when analyzing data with a complicated structure.

CHAPTERS III

Parallel Model Training of HGLLiM

Model building time is a critical issue for large-scale and complex datasets. Under the HGLLiM framework, as the number of samples increases, the time it takes to compute the posterior probability in the expectation step increases. In addition, it takes longer for the EM algorithm to converge, and it becomes more difficult to find a proper initial setting as the structure of the dataset becomes more complicated. To accelerate the model building process, we propose parallel HGLLiM, which is an extension of HGLLiM utilizing the hierarchical structure for parallel training. We use two complicated large-scale datasets to demonstrate the usage of the parallelization technique and the power of HGLLiM for modeling complex associations over a large number of observations.

3.1 Parallel model training of large-scale datasets

Parallel HGLLiM starts by subsetting the dataset into smaller groups. With the hierarchical model structure of HGLLiM, the model can be broken down into smaller sub-models and these sub-models can be trained separately. The hierarchical model structure also provides a simple way to aggregate sub-models back into the full model. HGLLiM determines local clusters using the low-dimensional data T. Thus, subsetting the dataset according to T would be a straightforward practice.

Categorical variables are suitable references for subsetting a dataset. However, the subsetting strategy is not limited to categorical variables, nor is it limited to T as well. As long as we can easily determine the group membership using X information, we could subset the dataset and accelerate the training procedure.

Once we determine the group membership, each group is trained separately. A group contains K global clusters and a global cluster contains up to M local clusters. When conducting the prediction, we should aggregate the model parameters from all groups. Denote $\theta_g = \{c_{gkl}, \Gamma_{gkl}, \rho_{gkl}, A_{gkl}, b_{gkl}, \Sigma_{gk}\}_{k=1,l=1}^{K_g,M_g}$ as the model parameter for group g; K_g, M_g are the corresponding global/local cluster numbers with $g = \{1, ..., G\}$. The aggregated HGLLiM model parameter can be expressed as $\theta = \{\theta_g\}_{g=1}^{G}$. Let V be the latent variable for group assignment. We obtain:

$$p(X = x|Y = y, V = g, Z = k, U = l; \theta) = \mathcal{N}(x; A_{gkl}y + b_{gkl}, \Sigma_{gk}) \tag{3.1}$$

The hierarchical model structure can be completed by

$$p(Y = y|V = g, Z = k, U = l; \theta) = \mathcal{N}(y; c_{gkl}, \Gamma_{gkl}), \tag{3.2}$$

$$p(V = g, Z = k, U = l; \theta) = \rho_{gkl}\frac{N_g}{N} = \phi_{gkl}, \tag{3.3}$$

where N_g is the number of samples within group g and $N = \sum_{g=1}^{G} N_g$. The inverse and forward conditional density can be written as:

$$p(X = x|Y = y) =$$
$$\sum_{g=1}^{G}\sum_{k=1}^{K_g}\sum_{l=1}^{M_g}\frac{\phi_{gkl}\mathcal{N}(y; c_{gkl}, \Gamma_{gkl}; \theta)}{\sum_{i=1}^{G}\sum_{j=1}^{K_g}\sum_{m=1}^{M_g}\phi_{ijm}\mathcal{N}(y; c_{ijm}, \Gamma_{ijm}; \theta)}\mathcal{N}(x; A_{gkl}y + b_{gkl}, \Sigma_{gk}; \theta), \tag{3.4}$$

$$p(Y = y|X = x) =$$
$$\sum_{g=1}^{G}\sum_{k=1}^{K_g}\sum_{l=1}^{M_g}\frac{\phi^*_{gk}\mathcal{N}(x; c^*_{gk}, \Gamma^*_{gk}; \theta^*)}{\sum_{i=1}^{G}\sum_{j=1}^{K_g}\sum_{m=1}^{M_g}\phi^*_{ijm}\mathcal{N}(x; c^*_{ijm}, \Gamma^*_{ijm}; \theta^*)}\mathcal{N}(y; A^*_{gkl}x + b^*_{gkl}, \Sigma^*_{gkl}; \theta^*), \tag{3.5}$$

where $\theta^* = \{\theta^*_g\}_{g=1}^G$ and θ^*_g is the forward model parameter that corresponds to θ_g. The prediction can be conducted by

$$\mathbb{E}[Y|X = x] = \sum_{g=1}^{G}\sum_{k=1}^{K_g}\sum_{l=1}^{M_g}\frac{\phi^*_{gkl}\mathcal{N}(x; c^*_{gkl}, \Gamma^*_{gkl}; \theta^*)}{\sum_{i=1}^{G}\sum_{j=1}^{K_g}\sum_{m=1}^{M_g}\phi^*_{ijm}\mathcal{N}(x; c^*_{ijm}, \Gamma^*_{ijm}; \theta^*)}(A^*_{gkl}x + b^*_{gkl}). \tag{3.6}$$

Note that for testing data x_n, the probability of being assigned to group g, global cluster k and local cluster l is

$$r_{ngkl} = \frac{\phi^*_{gkl}\mathcal{N}(x_n; c^*_{gkl}, \Gamma^*_{gkl}; \theta^*)}{\sum_{i=1}^{G}\sum_{j=1}^{K_g}\sum_{m=1}^{M_g}\phi^*_{ijm}\mathcal{N}(x_n; c^*_{ijm}, \Gamma^*_{ijm}; \theta^*)}. \tag{3.7}$$

3.2 Magnetic resonance vascular fingerprinting

It is of great interest to the scientific community to be able to efficiently assess microvascular properties, such as blood volume fraction, vessel diameter, and blood oxygenation the in brain so that the ability to diagnose and manage brain diseases can be improved. Recently, a new approach called magnetic resonance vascular fingerprinting (MRvF) was proposed as an alternative to overcome the limitations of

analytical methods in measuring microvascular properties. The approach was built on a system in which "fingerprints" in every voxel are compared to a dictionary obtained from numerical simulations (Ma et al., 2013). Finding the closest match to a fingerprint record in the dictionary allows a direct link between the parameters of the simulations and the microvascular variable (also referred to as a "parameter" in these studies). In this first approach of Ma et al. (2013), the authors use the nearest neighbor search in Euclidian distance to find the match. In Lemasson et al. (2016), the investigators simulate MR signals from a virtual voxel. The parameter inputs of the simulations, including blood volume fraction and mean vessel radius, among others, are varied to construct a dictionary of possible signal evolutions. They also use the nearest neighbor search, though not with Euclidian distance, to identify a match. The goal of our study is to build a model-based system that carries out the same prediction tasks these investigators are interested in: taking a high-dimensional fingerprint as an input and predicting the low-dimensional parameter(s) as the outcome. In addition, the model we construct would assist investigators in having a better understanding of the complexity of the system. The existing nearest neighbor search/match methods do not help to understand the association and the underlying structure of the system. Furthermore, it is known that a nearest neighbor prediction approach may not work that well for data bearing high variation noises, a phenomenon we illustrate empirically in Section 3.2.2. In Lemasson et al. (2016), their search/match procedure is effective in predicting a parameter, Blood Volume Fraction (BVf), at an aggregated level. Precisely, the true and predicted means of BVf over voxels of a given region are practically the same. However, the performances of the method in predicting another parameter, Apparent Diffusion Coefficient (ADC, not discussed in the published work), at the voxel level, are less ideal.

A synthetic magnetic resonance vascular fingerprint (hereafter referred to as fingerprint) dataset composed of $1,383,648$ observations was created to serve as a

"search/match" library. Each observation in the library consists of a fingerprint measurement and associated parameters: mean vessel radius (Radius), Blood Volume Fraction (BVf) and a measurement of blood oxygenation (DeltaChi). One goal is to predict these parameters ($L_t = 3$) using the fingerprint measurement ($D = 32$). In addition to these three parameters, other parameters (variables) that have influence over the fingerprint measurements include Apparent Diffusion Coefficient (ADC), vessel direction (Dir) and vessel geometry (Geo).

3.2.1 Analysis and subsetting of the synthetic data

To speed up the model building procedure, we take advantage of the subsetting and parallelization technique described in Section 3.1. The synthetic dataset is analyzed and subsetted based on the analysis in this section. In Table 3.1, we summarize the values and the range of the microvascular parameters ($t_1 \sim t_6$) of the fingerprint dictionary. The values for each parameter are shown in Figure 3.1. There are 1,383,648 observations in the synthetic dictionary. The dataset is divided to cover as many kinds of data as possible for cross-validation purposes. First, we use t_6 to form Group 1 ($t_6 = 1$) and Group 2 ($t_6 = 2$). Our exploratory analysis shows high complexity when $t_6 = 3$. Thus, it is necessary to separate more groups on $t_6 = 3$ to reflect the complexity. For data with $t_6 = 3$, we divide t_1 into 3 categories and consider 6 different values in t_5. All together, for $t_6 = 3$, we construct 18 groups (Group 3 to Group 20). The available size of each group is shown in Table 3.2.

To construct 20-fold cross-validation, the testing sample size is picked so that all data within the smallest group would be used. The smallest group size is 2030 (Group 3 to Group 14). Within these groups, we select 102 testing samples from each group. Some data could have replicates, but the number of replicates would be no more than 2. This aims to make the number consistent through all groups and folds. After excluding testing data, we randomly pick 10,000 for Group 1 and Group 2 as training

Parameter	Parameter meaning	No. of unique values	Range
t_1	R (μm)	38	$0.5 \sim 1000$
t_2	BV (%)	47	$0.25 \sim 50$
t_3	ADC ($\mu m \cdot s^{-1}$)	33	$2 \times 10^{-10} \sim 18 \times 10^{-10}$
t_4	DeltaChi (ppm)	29	$0 \sim 1.4$
t_5	Direction (radians)	6	0, 0.314, 0.628, 0.943, 1.257, 1.571
t_6	Geometry	3	1, 2, 3

Table 3.1: The unique values and the range of microvascular parameters

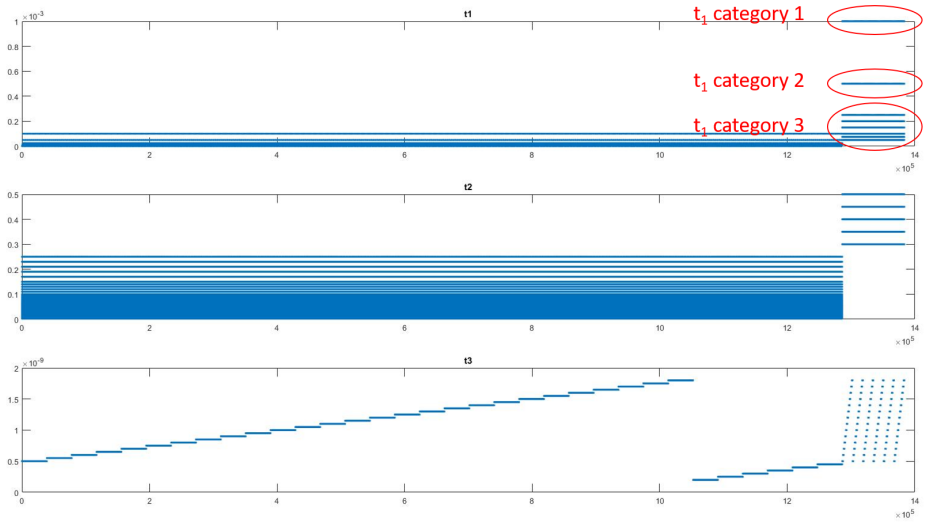
Group ID	Value	Available Size
Group1	$t_6 = 1$	1052352
Group2	$t_6 = 2$	233856
Group3	t_1 category1, t_5 value1	2030
Group4	t_1 category1, t_5 value2	2030
Group5	t_1 category1, t_5 value3	2030
Group6	t_1 category1, t_5 value4	2030
Group7	t_1 category1, t_5 value5	2030
Group8	t_1 category1, t_5 value6	2030
Group9	t_1 category2, t_5 value1	2030
Group10	t_1 category2, t_5 value2	2030
Group11	t_1 category2, t_5 value3	2030
Group12	t_1 category2, t_5 value4	2030
Group13	t_1 category2, t_5 value5	2030
Group14	t_1 category2, t_5 value6	2030
Group15	t_1 category3, t_5 value1	12180
Group16	t_1 category3, t_5 value2	12180
Group17	t_1 category3, t_5 value3	12180
Group18	t_1 category3, t_5 value4	12180
Group19	t_1 category3, t_5 value5	12180
Group20	t_1 category3, t_5 value6	12180

Table 3.2: The size of each group of the fingerprint dataset.

samples. For Group 3 to Group 14, the remaining 1928 samples would become the training data. For Group 15 to Group 20, we pick 2000 training samples. As a result, within each fold, there would be 55136 training samples (10,000 from Group 1 and Group 2, 1928 from Group 3 to Group 14, 2000 from Group 15 to Group 20) and 2040 testing data (102 from each group).

For HGLLiM and GLLiM-structure, it takes about 549.86 and 362.94 seconds, re-

(a)



(b)

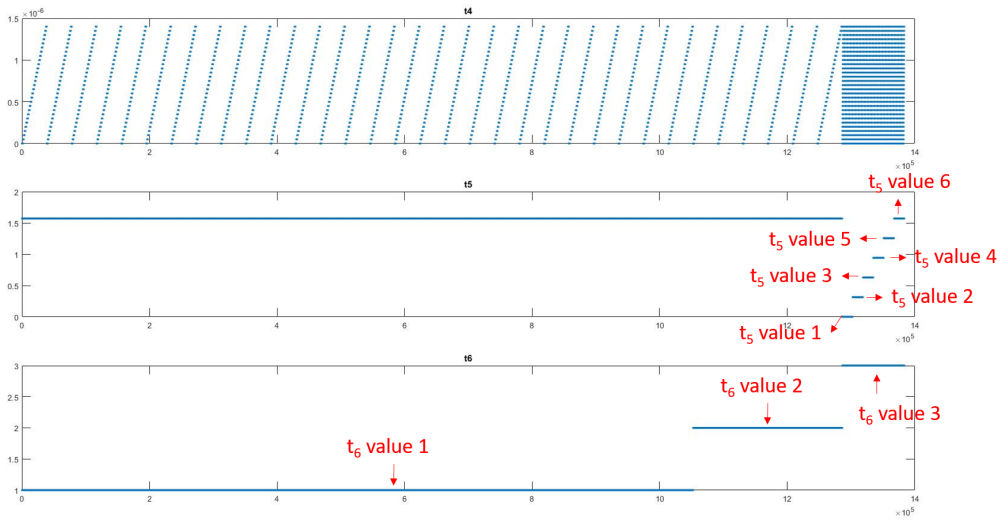


Figure 3.1: The distribution of parameters (T). The x-axis shows the index of observations and the y-axis marks the values of each observation in different dimensions. (a) Dimensions 1 to 3; (b) Dimensions 4 to 6.

spectively, for each method to complete the EM computation. In comparison, it takes 19341.51 and 14107.63 seconds without using the parallel computing strategy. We evaluate and compare the performance of different methods through cross-validation and show that the model-based methods can achieve comparable results.

3.2.2 Numerical results

Our current study consists of two components. Through cross-validation, we first evaluate the feasibility and effectiveness of the parallel computation algorithms and compare the performance of different methods on the synthetic dataset. We then apply these methods to a fingerprint dataset collected from an animal study; in which, besides predicting the variable BVf (the main goal of Lemasson et al. (2016)), we focus on predicting another variable, ADC, a more challenging scenario which has not been reported before. The synthetic library is divided into 20 groups, and we apply the parallelization techniques to accelerate the model building process. When conducting the analysis of the animal study, we add a small amount of the *in vivo* data to the training dataset. We noted that fingerprint samples from the real world are noisier than their synthetic counterparts and thus this practice, as a calibration step, enables the training model to readily accommodate the real fingerprint samples in prediction. The ratio of the synthetic samples to the real image samples is 4 to 1. The cluster number and latent factor number are selected using the method described in Section 2.3.3 and are set to $K = 1240$ and $L_w = 9$. We evaluate and compare the performance of different methods on the synthetic dataset through cross-validation. The cross-validation results in predicting Radius, BVf and DeltaChi demonstrate that the model-based methods (GLLiM/HGLLiM/GLLiM-structure) can achieve comparative prediction performance. Next, we apply these methods to a fingerprint data set collected from an animal study.

In Lemasson et al. (2016), numerical performances of a dictionary matching method were presented. For comparison purposes, we implement the dictionary matching method adopted in Lemasson et al. (2016). The coefficient of determination (r^2) is used to measure the similarity between a testing sample and the training samples (dictionary). The coefficient of determination, r^2, between a testing sample $x^{test} \in \mathbb{R}^D$ and a training sample $x^{train} \in \mathbb{R}^D$ is calculated as:

$$r^2 = 1 - \frac{\sum_{d=1}^{D}(x_d^{test} - x_d^{train})^2}{\sum_{d=1}^{D}(x_d^{test} - \bar{x}^{test})^2}, \tag{3.8}$$

where $\bar{x}^{test} = \frac{1}{D}\sum_{d=1}^{D} x_d^{test}$ and the subscript d is used to denote the index of the dimension. The matched fingerprint is the training fingerprint with the largest r^2 and we predict the parameters of the testing data using the matched fingerprint.

Cross-validation results for the synthetic fingerprint dataset

Table 3.3 shows the 50%, 90% and 99% quantiles of the prediction squared errors for different parameters using different methods through cross-validation. The outcomes reported under the 50th and 90th percentiles give the indication of "average" and "almost-all" prediction performances for each method. The 99th percentile values allow the comparisons of worse-case scenarios. We observe that the prediction is close to the true value for 90% of the predicted values. Using GLLiM, we obtain slightly larger values of E^2 for Radius. However, all four methods reach similar values of E^2 for Radius at the 99% quantile. For BVf, GLLiM performs worse than other methods but its 99% squared error level is still acceptable. The prediction performances of BVf for all methods are better than those of other parameters, with the relationship between BVf and Y being the strongest among all parameters. For DeltaChi, the E^2's for dictionary matching are larger than those of other methods at the 90% quantile level. At the 99% quantile level, its performances become similar to those of HGLLiM and GLLiM-structure. Note that the model is built using Radius, BVf, ADC and DeltaChi. The parameter ADC is included to evaluate the prediction performance on real image data. However, note that adding a weakly informative parameter such as ADC to the model would downgrade the prediction performance. If predicting ADC is not the major task, we could obtain lower prediction error when training the model with Radius, BVf and DeltaChi. The results of using 3 parameters are shown in Table 3.4.

On the other hand, when adopting dictionary matching, testing data are compared to fingerprint observations, which are associated with 6 parameters as shown in Figure 3.1. With all parameters embedded inside fingerprint observations, the dictionary matching method actually uses information from 6 parameters. If we restrict the parameter space, i.e. only consider Radius, BVf and DeltaChi, there would be multiple fingerprints associated with the same set of the restricted parameters. To evaluate the performance under a restricted parameter setting, we randomly select a fingerprint as the representative for the same set of parameters. Table 3.4 shows the cross-validation results on the restricted synthetic fingerprint dataset.

Comparing Table 3.4 to Table 3.3, we observe improvement on 90% quantiles for model-based methods, which indicates that we could obtain better prediction outcomes by removing ADC from the training data. In contrast, the results of the dictionary matching method become worse. This is a natural consequence of lacking sufficient details to categorize and distinguish samples in the dictionary. If the remaining parameters are insufficient to reflect the data complexity, testing data will likely be matched to an inadequate member within the dictionary and, as a result, we would obtain a large prediction error. This comparison shows the difference between the dictionary matching method and the model-based method. For the dictionary matching method, we hope to enumerate all possible distinctions in the dictionary. Thus, the prediction performance deteriorates when this goal cannot be achieved. However, this may not apply to model-based methods, where the most appropriate model among the ones being considered is used to conduct prediction. The performance could improve when weakly informative parameter covariates are removed.

The animal study dataset contains samples from 115 rats categorized into 5 different groups: healthy, 3 kinds of tumors (9L, C6 and F98) and stroke. For each rat, there are 5 brain slices of 128×128 voxels and each voxel contains 32-dimension fingerprint information. For each slice, the lesion (unhealthy) and the striatum (healthy)

	Dictionary matching			GLLiM			HGLLiM			GLLiM-structure		
	50%	90%	99%	50%	90%	99%	50%	90%	99%	50%	90%	99%
Radius	$< 10^{-4}$	0.2843	21.44	$< 10^{-4}$	0.3114	21.44	$< 10^{-4}$	0.2144	21.44	$< 10^{-4}$	0.2144	21.44
BVf	$< 10^{-4}$	$< 10^{-4}$	0.0023	$< 10^{-4}$	$< 10^{-4}$	0.0406	$< 10^{-4}$	$< 10^{-4}$	0.0091	$< 10^{-4}$	$< 10^{-4}$	0.0242
DeltaChi	$< 10^{-4}$	0.0143	0.3571	$< 10^{-4}$	0.0132	0.5972	$< 10^{-4}$	0.0009	0.2236	$< 10^{-4}$	0.0007	0.3361

Table 3.3: The 50%, 90% and 99% quantiles of squared error using different methods. The models are built upon 4 microvascular parameters: Radius, BVf, ADC, and DeltaChi.

	Dictionary Matching			GLLiM			HGLLiM			GLLiM-structure		
	50%	90%	99%	50%	90%	99%	50%	90%	99%	50%	90%	99%
Radius	0.2144	69.3636	82.5297	$< 10^{-4}$	0.2916	21.44	$< 10^{-4}$	0.2144	21.44	$< 10^{-4}$	0.2144	21.44
BVf	$< 10^{-4}$	0.2277	0.2277	$< 10^{-4}$	$< 10^{-4}$	0.0261	$< 10^{-4}$	$< 10^{-4}$	0.0068	$< 10^{-4}$	$< 10^{-4}$	0.0269
DeltaChi	$< 10^{-4}$	0.0571	0.7000	$< 10^{-4}$	0.0108	0.6385	$< 10^{-4}$	0.0013	0.2012	$< 10^{-4}$	0.0005	0.3158

Table 3.4: The 50%, 90% and 99% quantiles of squared error using different methods. The models are built upon 4 microvascular parameters: Radius, BVf and DeltaChi.

areas are labeled, and they form the region of interest (ROI). Figure 3.2 shows the predicted BVf image using different methods. As indicated in Lemasson et al. (2016), the values of true BVf are not available at the voxel level, and instead, they are measured over the whole ROI. Nevertheless, the comparison between the true values and those obtained by the dictionary matching method, at the ROI level, indicates that the method has successfully provided a close-to-truth match; see Lemasson et al. (2016). Table 3.5 shows the mean prediction results within the ROI's obtained by different methods. The three additional methods considered here, besides the dictionary matching method used in Lemasson et al. (2016), are GLLiM, HGLLiM and GLLiM-structure. All four methods provide similar results in predicting BVf.

There are 1,385,509 samples in the real image dataset. For the dictionary matching method, using a parallel for-loop (*parfor*) and a pre-processing technique (Lemasson et al. (2016)), it took about 2.4 hours (precisely 8639.53 seconds) to match the whole animal image samples to the training dataset ($N^{train} = 1,383,648$). A direct computation without *parfor* and pre-processing took 429507.79 seconds and reached the same outcomes. For the model-based method, utilizing the grouped structure and the parallel computing technique, it takes 1058.32/2133.51/1922.37 seconds for

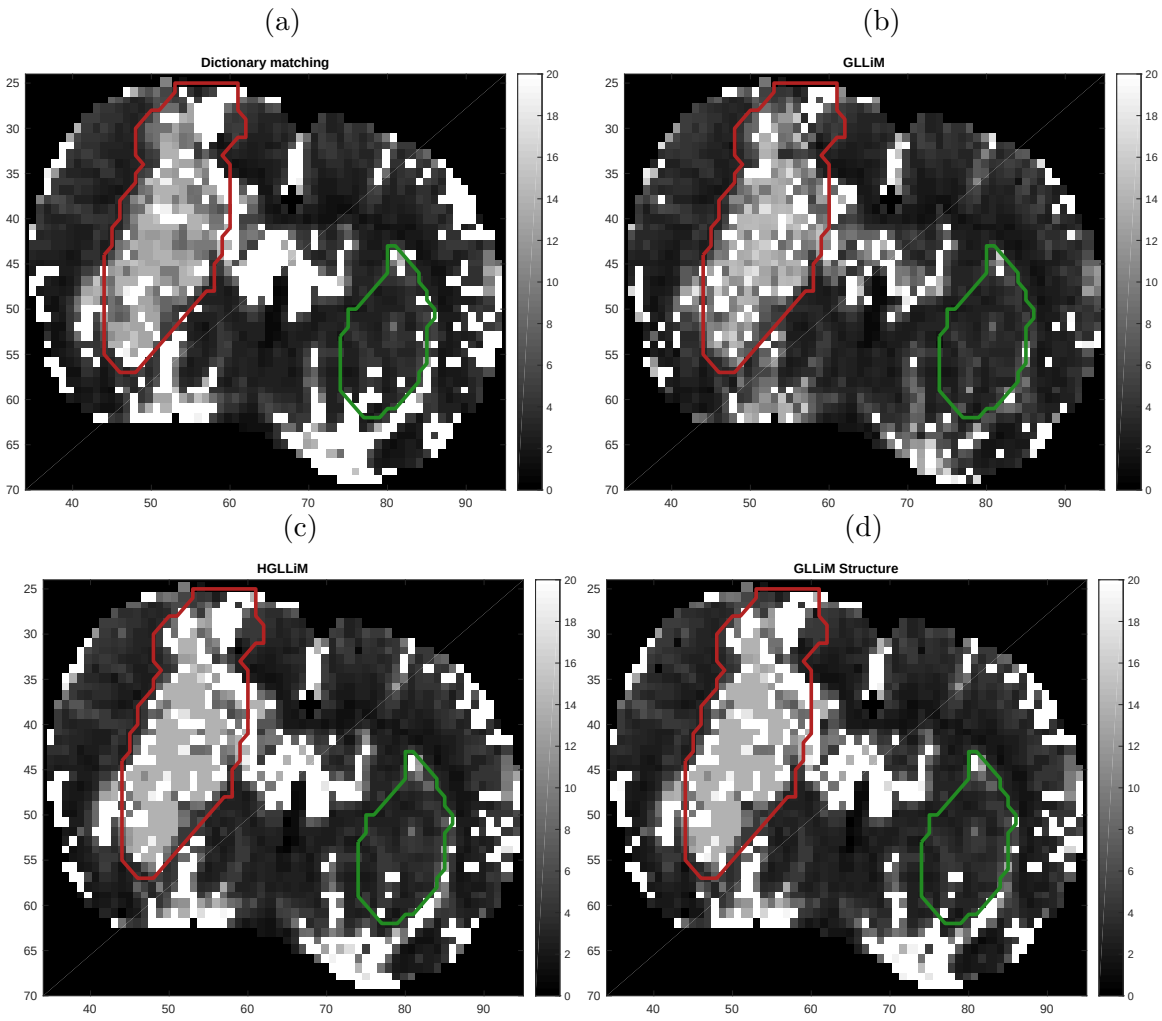


Figure 3.2: The predicted BVf images of one animal from the 9L group using either (a) dictionary matching, (b) GLLiM, (c) HGLLiM or (d) GLLiM-structure. In each plot, the ROI on the left marks the lesion region and the ROI on the right is from the healthy striatum.

GLLiM/HGLLiM/GLLiM-structure to process the animal image dataset. Thus, the prediction procedure of GLLiM/HGLLiM/GLLiM-structure is much more efficient than the dictionary matching method.

The parameter ADC was not thoroughly investigated in Lemasson et al. (2016). The main reason is that the predicted ADC values, obtained using the dictionary matching approach, were not comparable to the ones measured *in vivo* by MRI. With the *in vivo* ADC values available at the voxel-level, being able to understand how the synthetic and real measurements differ for a given parameter is scientifically important

		Dictionary matching	GLLiM	HGLLiM	GLLiM-structure
9L	Radius	21.85	20.14	22.12	21.52
	BVf	14.49	14.33	14.71	14.25
	DeltaChi	0.98	0.93	1.03	0.94
C6	Radius	13.59	16.01	13.67	13.81
	BVf	4.17	4.01	4.25	4.52
	DeltaChi	0.77	0.76	0.79	0.74
F98	Radius	11.56	13.14	11.13	11.23
	BVf	3.86	3.96	4.01	3.97
	DeltaChi	0.65	0.66	0.62	0.61
Stroke	Radius	14.69	13.51	14.31	14.41
	BVf	4.22	4.49	4.13	4.25
	DeltaChi	0.60	0.63	0.62	0.63
Healthy	Radius	8.16	7.96	8.54	8.34
	BVf	3.58	3.51	3.63	3.56
	DeltaChi	0.76	0.72	0.74	0.80

Table 3.5: The mean predicted values within ROIs of different vascular parameters from different categories.

to developing new instruments and to future knowledge advancements. Here, we study ADC and use it to evaluate the prediction performances of different methods. Figure 3.3 shows the true ADC image and the images of the differences between the true and predicted ADC values. The differences are shown in the ratio against the signal levels for each ROI. Most of the predictions made by dictionary matching deviate from the true values. On the other hand, HGLLiM and GLLiM-structure provide better ADC images. There are some voxels with extreme differences (dark red or dark blue) that all methods cannot predict well. When no suitable training information can be provided by the synthetic fingerprint data, the prediction quality on these voxels tends to be dreadful regardless of which method is used.

Table 4.11 shows the 50%, 90% and 99% quantiles of the ADC squared errors. We still obtain some predictions with large errors using GLLiM/HGLLiM/GLLiM-structure. However, for the majority of the data, the squared errors are smaller than

	Dictionary matching			GLLiM			HGLLiM			GLLiM-structure		
	50%	90%	99%	50%	90%	99%	50%	90%	99%	50%	90%	99%
9L	1.1180	3.9803	10.6829	0.2392	0.5684	14.5668	0.1132	0.7613	11.8721	0.1018	0.7154	10.9574
C6	1.1208	4.4719	14.4888	0.3043	2.6091	26.7575	0.3252	1.9840	22.5427	0.3138	1.7764	20.0213
F98	1.0994	4.2373	14.4888	0.3802	3.4129	55.4479	0.2951	2.3672	35.3199	0.2801	2.4129	50.8133
Stroke	1.1663	5.8045	14.8888	0.4779	4.5668	66.1164	0.3218	3.0975	55.7821	0.3192	3.1424	53.9315
Health	1.0931	3.8086	7.7912	0.2131	1.2510	14.5668	0.1527	1.1087	11.9597	0.1054	1.1145	13.2165

Table 3.6: The 50%, 90% 99% quantiles of ADC squared errors for different methods on different image categories.

those obtained by the dictionary matching method. We determine that there is no suitable cluster to conduct prediction for these data. For GLLiM/HGLLiM/GLLiM-structure, if a suitable cluster for conducting prediction does not exist, the cluster with the closest Mahalanobis distance is applied for prediction. However, the largest membership posterior probability r_{ngkl} among all g, k, l in Equation (3.7) would be smaller than the majority of the data. This information could be utilized to identify unreliable prediction results. The worst case of dictionary matching seems to produce smaller prediction error when being compared to other methods. Nevertheless, this is due to the nature of the difference among approaches. The dictionary matching method always makes predictions using values obtained from a member in the dictionary, so that its prediction error cannot go beyond what would be provided by the possible values in the dictionary. This phenomenon does not apply to model-based methods. When the prediction is conducted on data outside of the range of the training dataset, the prediction error could become considerably large, as shown by the outcome of 99 percentiles of prediction squared errors. As a result, even though dictionary matching seems to outperform other model-based methods in these extreme cases, it does not necessarily indicate that the dictionary method is practically useful for these cases, with the outcomes being so much worse when predicting the rest of the dataset. Our model-based approaches, on the other hand, do have the advantage of identifying these troublesome cases for further consideration.

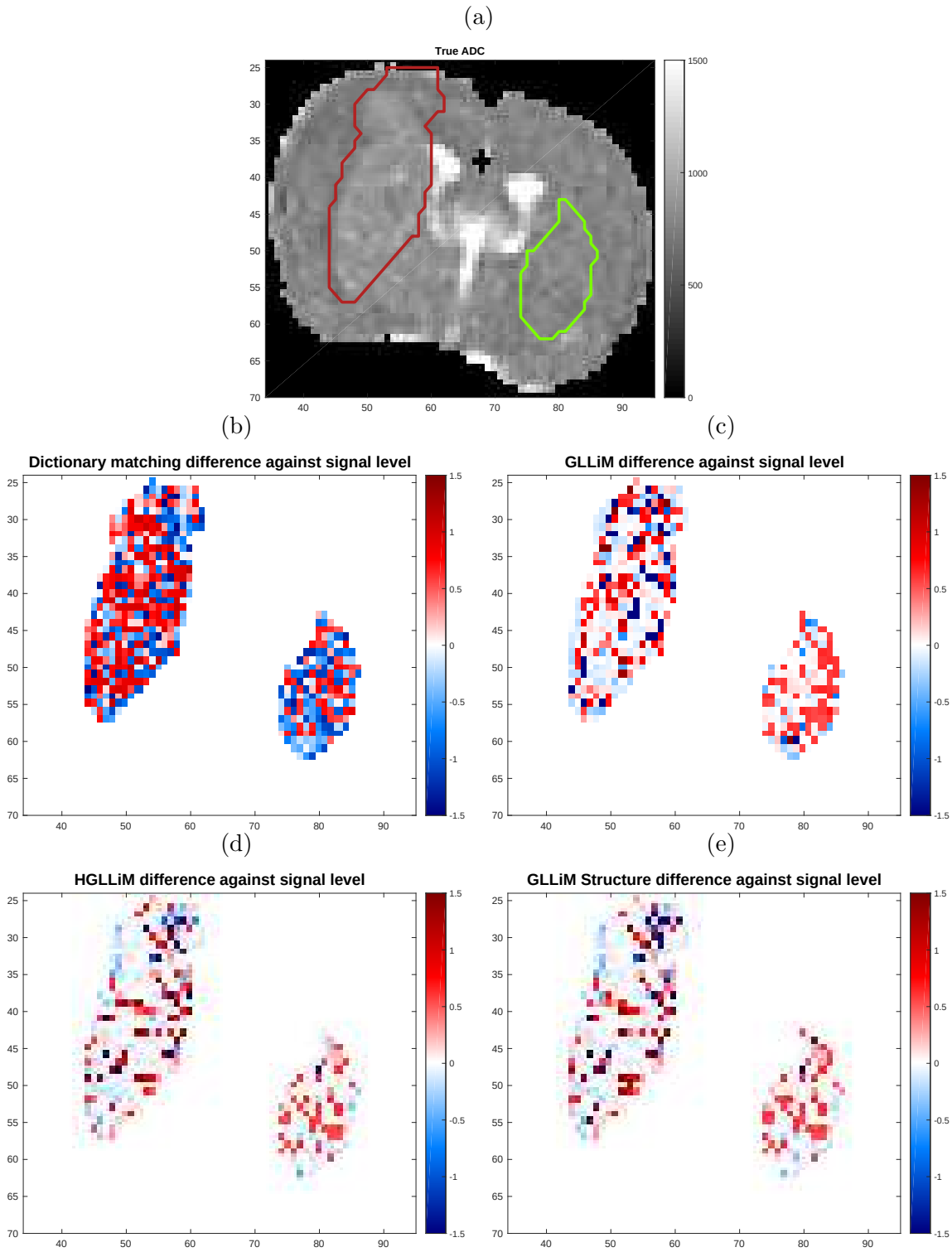


Figure 3.3: The true ADC images and the differences between the true values and the predicted values against the signal levels of one animal from the 9L group. Differences are normalized by the average true ADC values in each ROI. (a) The true ADC image. Difference maps between true values and predicted values against the signal levels using either (b) the dictionary matching method, (c) GLLiM, (d) HGLLiM or (e) GLLiM-structure.

3.3 Single-channel source separation

Single-channel source separation (SCSS) aims to separate different sources from a mix of sound sources. As an example, a sound mix could contain a singing voice and a musical accompaniment as shown in Figure 3.4. The goal of SCSS is to extract the singing voice and the musical accompaniment using the given mixture. The task becomes even more challenging if one wants to separate the musical accompaniment into different instruments such as piano, guitar, drums, etc.

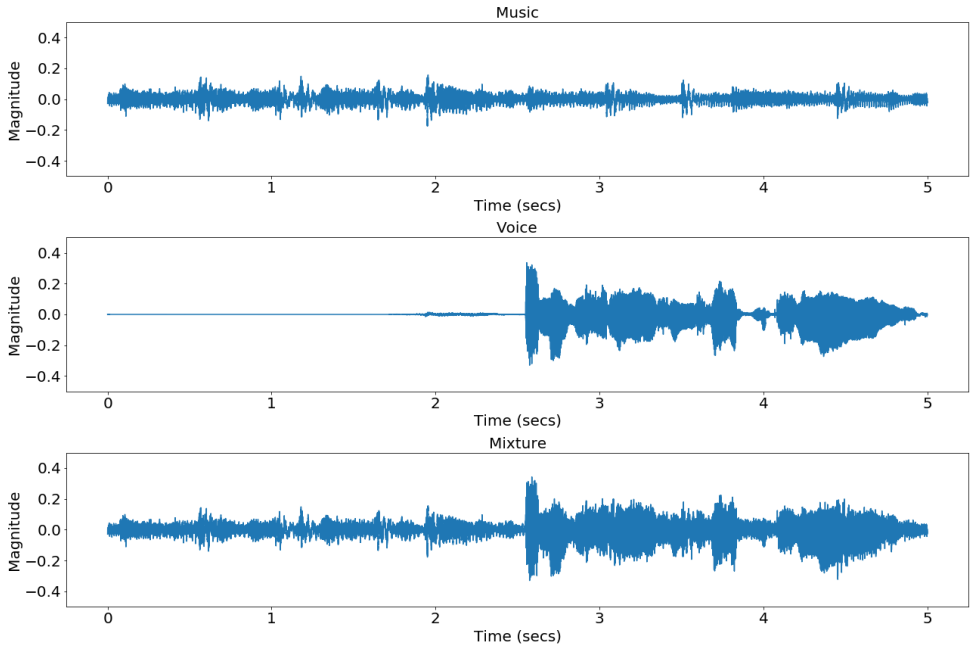


Figure 3.4: An example of the waveform of the music, the voice and their mix.

SCSS can be treated as a preprocessing procedure of other algorithms. By separating sound sources, one can apply different techniques to different sound sources and obtain a better sound effect. Nowadays, audio data are often stored in mono or stereo format, i.e., one or two sound channels. However, people usually own equipment with more than two channels. The task of upmixing is to determine the transformation from a mono or stereo sound source to a system with a higher number of channels, and SCSS is an important preprocessing step for the mixing algorithm (Fitzgerald, 2011). Furthermore, the accuracy of automatic speech recognition can be improved

by isolating the voice source from the background noise (Maas et al., 2012). Similarly, the performance of music information retrieval can be improved when different types of instruments are separated (Hsu and Jang, 2010).

SCSS is a challenging problem since only one mixed sound source is given and one has to generate more than one different sources. A great amount of effort has been made to address the problem. Several approaches adopt low-rank approximation (Sprechmann et al., 2012; Yang, 2013). People adopt matrix factorization to find meaningful bases and weights for separation. Examples include non-negative matrix factorization (NMF) (Grais and Erdoğan, 2012; Schmidt and Olsson, 2006; Virtanen, 2007) and PCA (Huang et al., 2012). However, this assumption may not always be valid, and thus the separated results may not be satisfactory. Recently, neural networks have drawn plenty of attention. It is known that neural networks can approximate a wide variety of functions (Hornik et al., 1989). For the problem of SCSS, people utilize different structures of neural networks that can effectively separate sound sources. The use of the feed-forward network is based on the universal approximation theorem (Grais et al., 2016; Narayanan and Wang, 2013). It shows the capability of modeling the complicated relationship between a mixed sound source and individual ones. However, this type of network requires many parameters and could be easily overfitted. A convolutional neural network (CNN) adopts shared-weights architecture (Grais and Plumbley, 2017; Simpson et al., 2015), and thus the number of parameters is smaller. However, CNN assumes shift invariant, and thus it cannot capture the complicated time correlation between audio samples. A recurrent neural network (RNN) models the temporal behavior between neighboring audio samples (Huang et al., 2014, 2015; Maas et al., 2012). Nevertheless, the complex structure of RNN makes it harder to converge. All of the neural network-based methods are subject to similar issues: tuning. There are many hyperparameters such as the number of layers, number of neurons, etc. The choice of activation types, the

choice of algorithm for optimization and plenty of other factors can affect the performance of neural networks. Thus, it is time-consuming to figure out the appropriate combination of settings that could give satisfactory results.

In this work, we seek model-based approaches to separate the mixed sound sources into different ones. We adopt GLLiM and HGLLiM to model non-linear mappings between mixed sound sources and the time-frequency masks. In the following sections, we first describe the details of constructing the time-frequency mask. Next, we explain how locally linear mappings can be applied to the single-channel source separation problem using the subsetting and parallelization technique. Finally, we compare the prediction results of the model-based methods to the results obtained from neural networks.

3.3.1 Time-frequency masking

Time-frequency masking is a widely adopted approach for source separation (Grais et al., 2016; Huang et al., 2014; Simpson et al., 2015; Weninger et al., 2014). How humans perceive audio in the frequency domain has been extensively studied. The audible frequency range of the human ear is roughly 20 Hz to 20k Hz. This indicates that we only have to focus on the data within this frequency range. In addition, the frequency range of the human voice is approximately 300 Hertz to 3000 Hertz. Thus, when dealing with the voice signal, the data outside of this range can be eliminated to reduce model variance.

However, processing the data on the pure frequency domain would neglect the correlation between neighboring audio samples in the time domain. Thus, people usually transform audio data into the time-frequency domain so that we can analyze the frequency changes through different time frames. Short-time Fourier Transform (STFT) is a widely used tool to analyze sound data.

Short-time Fourier Transform (Allen, 1982) is a Fourier-based analysis that ex-

tracts local frequency features. It differs from the traditional Fourier transform by applying the transformation to different time frames. The audio data can be divided into different segments and Fourier transform can be applied. Since different segments come from different time frames, we can observe the changes over different frequencies and different time points. In practice, the time signal would be divided into overlapped frames to reduce the boundary effect. Let $x(t)$ denote the audio signal in a discrete time domain indexed by time t. The STFT of $x(t)$ can be expressed as

$$X(m, f) = \sum_{t=-\infty}^{\infty} x(t)w(t - m)e^{-j2\pi ft}, \tag{3.9}$$

where m is the quantized time variable and f is the quantized frequency variable. The window function $w(\cdot)$ is also discrete and is usually a Hanning window or a Gaussian window. To visualize the changes over time and frequency, one often draws the spectrogram with the squared magnitude of the STFT results:

$$S(m, f) = |X(m, f)|^2. \tag{3.10}$$

Figure 3.5 illustrates the spectrogram of the data shown in Figure 3.4 with the window size equal to 1024 points.

The problem of source separation can be formulated as follows. Given the mixture signal $x(t)$, which is a mix of two sound sources $s_1(t)$ and $s_2(t)$ such that $x(t) = s_1(t) + s_2(t)$, let $X(m, f)$ be the STFT of $x(t)$. We formulate the separation as:

$$X(m, f) = S_1(m, f) + S_2(m, f), \tag{3.11}$$

where $S_1(m, f)$ and $S_2(m, f)$ are two unknown STFTs of the separated sources. In this framework the difference of phase angles is ignored. Thus, the magnitude of the

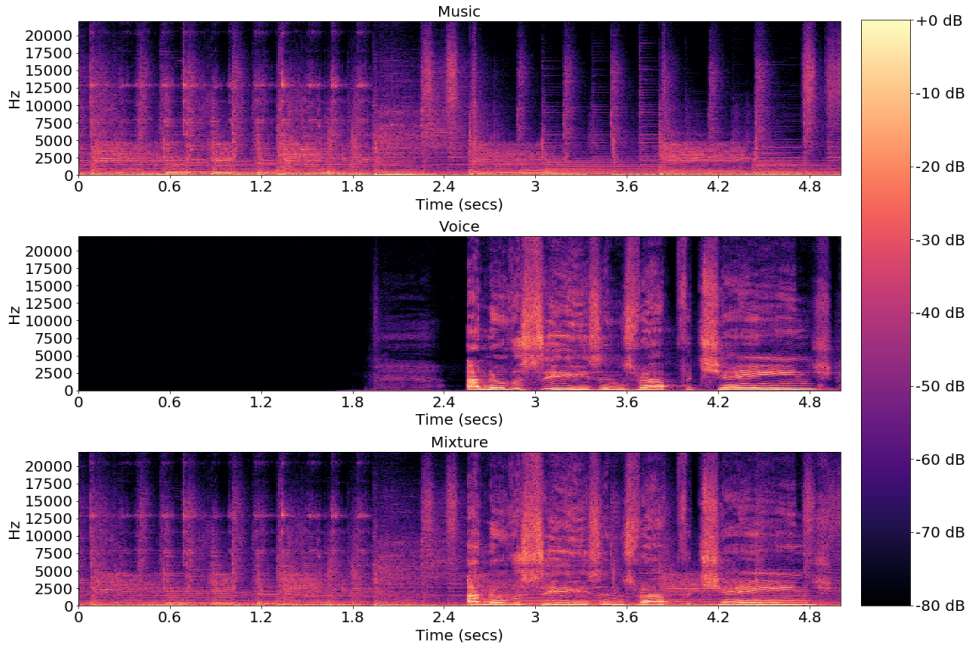


Figure 3.5: The spectrogram of the signals shown in Figure 3.4.

mixture spectrum can be approximated by the sum of the magnitude of the sources:

$$|X(m,f)| \approx |S_1(m,f)| + |S_2(m,f)|. \tag{3.12}$$

The notation can be simplified when expressing the spectrum with the matrix form: $|X| = |S_1| + |S_2|$. A time-frequency mask $M \in [0,1]$ is a matrix that scales the spectrum according to the contribution of different sources. We can obtain the STFT of sources using M by:

$$\hat{S}_1 = M \circ X, \tag{3.13}$$

$$\hat{S}_2 = (1 - M) \circ X, \tag{3.14}$$

where $\circ$ denotes element-wise multiplication and 1 is a matrix of ones. The main purpose is to find the M that can produce satisfactory separation. A common for-

mulation of M is the so-called soft mask:

$$M = \frac{S_1}{S_1 + S_2}, \tag{3.15}$$

where division and addition are both element-wise operations.

3.3.2 Using locally linear mappings for source separation

HGLLiM and GLLiM model the associations between the high-dimensional data X and the low-dimensional data Y. For the problem of source separation, we aim to predict the mask M time frame by time frame. That is, each column in the spectrum matrix is treated as a data point with the number of features equal to the number of Fourier transform points. We further split the mask across the frequency axis into disjoint subsets (banks) so that the mask in each bank is at low dimension. The goal of HGLLiM/GLLiM is to find the associations between the spectrum matrix and the mask values in a bank. We train the HGLLiM/GLLiM for different banks separately utilizing the parallelization technique. The separated models are aggregated together to perform prediction. Given two sources $s_1(t)$, $s_2(t)$ and their mixture $x(t)$, the training process is as follows:

1. Calculate the STFT of $s_1(t)$, $s_2(t)$ and $x(t)$. Denote the spectrum matrix as S_1, S_2 and X. Denote the size of these three spectrum matrices as $D \times N$, where D is the number of frequency bins and N is the number of samples.

2. Calculate $M = S_1/(S_1 + S_2)$, where division and addition are both element-wise operations.

3. Given a bank number G, separate M into banks along the frequency axis.

Denote Y_1 as the time-frequency mask in the first bank, we have:

$$Y_1 = \begin{bmatrix} m_1 \\ \vdots \\ m_B \end{bmatrix}, \tag{3.16}$$

where m_i denotes the i-th row of M and $B = \lceil \frac{D}{G} \rceil$ with $\lceil \cdot \rceil$ being the ceiling function. We can adopt a similar approach to construct Y_2, ..., Y_G.

4. Apply HGLLiM/GLLiM to construct models on (X, Y_1), (X, Y_2), ..., (X, Y_G).

The steps of predicting the time-frequency mask of the testing mixture $x^{test}(t)$ are as follows:

1. Calculate the STFT of $x^{test}(t)$. Denote the spectrum matrix as X^{test}.

2. Predict the time-frequency mask of each bank and combine them as M^{test}.

3. Apply mask values on X^{test} to obtain $\hat{S}_1^{test}$ and $\hat{S}_2^{test}$ (Equation (3.13), (3.14)).

4. Conduct inverse STFT on $\hat{S}_1^{test}$ and $\hat{S}_2^{test}$ to get $\hat{s}_1^{test}$ and $\hat{s}_2^{test}$.

3.4 Numerical results

The DSD100 dataset (Liutkus et al., 2017) contains 100 songs, which are split into 50 training songs and 50 testing songs. Each song is recorded in four different sources: vocals, bass, drums and others. Since our goal is to separate the mixed source into the singing voice (vocals) and the musical accompaniment, we mix the bass, drums and others into one soundtrack. This soundtrack is referred to as "music". The soundtracks of voice (vocals) and music are then mixed together. We apply STFT using the following settings: a Hanning window with 1024 points length and overlap interval of 512 points are used. The Fourier transform is taken at 1024 points,

and because of the symmetric property of the Fourier transform, the first 513 points are used. We clip every song into one-minute segments and adopt STFT analysis on these segments. The STFT result of each song (X) contains 5166 sample points. We note the dependencies between neighboring samples. To reduce the dependency, we only pick one STFT frame out of five when building the model. This leads to 51660 training samples for the whole training dataset. We split the dimension of T into 20 banks and thus $L_t = 26$ for the first 19 banks and $L_t = 19$ for the last one. The training process is accelerated with parallelization, which builds the model upon different songs and banks in parallel.

Cross-validation results on the training dataset

We apply GLLiM to each song for different settings of K and L_w for parameter selection. A 5-fold cross-validation is performed, and the average of the cross-validation testing prediction MSE over different banks is used to evaluate the settings. Figure 3.6 shows the cross-validation MSE (CV MSE) over 50 songs under different settings of K and L_w. Figure 3.6 shows the CV MSE under different settings of K and L_w. According to the cross-validation results averaged over 50 songs, $K = 5$ and $L_w = 15$ would be a suitable setting.

We also investigate the setting with larger values of L_w. Figure 3.7 shows the CV MSE for different L_w when $K = 5$. The CV MSE is 0.09195, 0.09006, 0.08979 and 0.0893 for $K = 15, 20, 25, 30$. That is, we obtain about 3% better prediction performance when increasing L_w from 15 to 30. However, increasing L_w from 15 to 30 would require more parameters, and thus even though we can obtain better CV MSE by setting larger values of L_w, we still set $L_w = 15$.

For GLLiM, the overall testing prediction MSE is 0.1172 when $K = 5$, $L_w = 15$. For HGLLiM, we simply set $M = 5$, $minSize = 5$, $dropThreshold = 0.5$ as suggested in Section 2.3.3 and let $K = 5$, $L_w = 15$. The testing prediction MSE is 0.0993, which

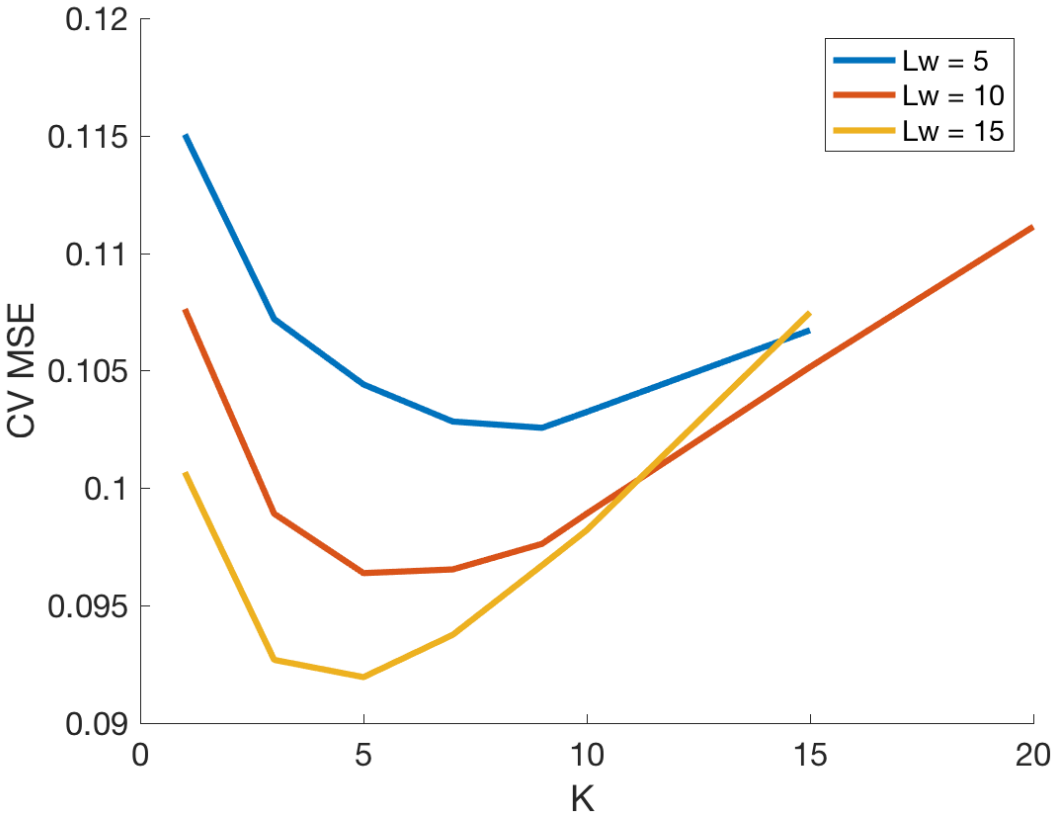


Figure 3.6: The average CV MSE for different settings of K and L_w

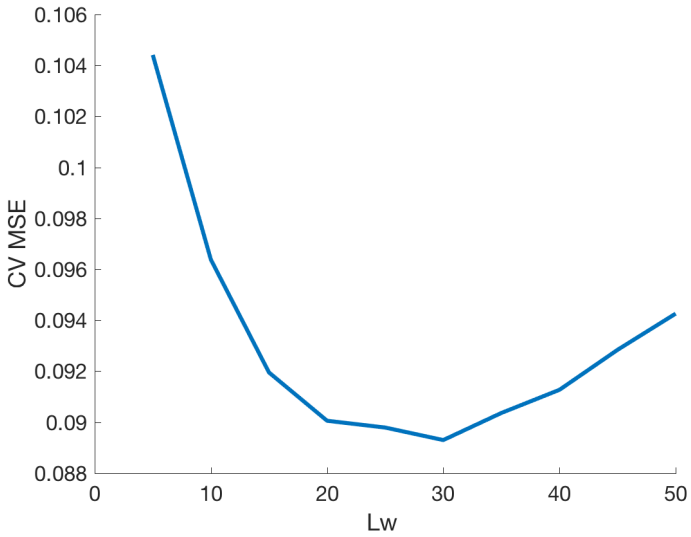


Figure 3.7: The average CV MSE for different L_w when $K = 5$

outperforms GLLiM by about 15%. Using the information learned by HGLLiM, the testing prediction MSE is 0.1002 for GLLiM-structure.

As a comparison, we implement the soft mask neural network in Grais et al. (2016). The input and output layers are of size 513. For the hidden layers, we construct three fully connected layers with size 513 as well. We train the network using the ADAM optimizer with the learning rate set to 10^{-4}. The model is trained for 1000 epochs. The testing prediction MSE is 0.0806.

For the neural network, it takes about 4 hours to train on an Nvidia Tesla K80 GPU. Note that K80 GPU is more than 10 times faster than CPU when training the neural network. When $K = 5$, $L_w = 15$, it takes about 68.4 seconds for GLLiM to train the model for a single bank. As for HGLLiM, under $K = 5$, $L_w = 15$ and $M = 5$, it takes 219.6 seconds for the model training. The training time for GLLiM-structure is about 187.5 seconds. If full parallelization is considered, the training process can be finished within 4 minutes. However, this would require using a lot of computing resources at the same time. If we parallelize the training process on each song and train different banks sequentially, the training process would take about 22.8/73.2/62.5 minutes for GLLiM/HGLLiM/GLLiM-structure. The training time of HGLLiM/GLLiM/GLLiM-structure is much faster compared to the training time of the neural networks.

3.5 Conclusion

Parallel model training is an effective technique to accelerate the training process. We present how the parallelization practice can be readily accommodated by HGLLiM's hierarchical model structure. Results of the fingerprint dataset show that HGLLiM can provide comparable prediction performance for one case and much smaller prediction errors for the other, compared to the dictionary matching method, in only 25% of the computational time. In the case of the sound dataset, parallel

HGLLiM can save about 70% of the training time compared to the neural networks.

CHAPTERS IV

Robust Gaussian Locally Linear Mapping

Mixtures of regressions (De Veaux, 1989), as a regression extension of the mixture of densities model, establish the associations between covariates and responses under the scenarios when mixtures exist. Complicated mappings can be broken down into simpler setups using mixtures, and combining direct linear associations established from mixtures can approximate the sophisticated non-linear relationship. Mixtures of regressions are widely used in different scientific fields. In speech engineering, identifying a mapping function from one feature space of the source to another feature space is known as *voice conversion*, which is a technique widely used for speech enhancement and language education for non-native speakers (Qiao and Minematsu, 2009; Stylianou et al., 1998; Toda et al., 2008; Zen et al., 2009). Other applications of mixture regression include age-identification, where one estimates a subject's age based on his/her facial image. The task is typically formulated as a nonlinear mapping problem that maps several facial features to different ages (Han et al., 2017; Huang et al., 2017). In modern geosciences, researchers are interested in recovering physical parameters using hyperspectral images (Bioucas-Dias et al., 2012; Brown et al., 2000). As an example, given remotely sensed data of Mars, researchers aim to estimate the chemical composition of the Mars surface (Deleforge et al., 2015). Mixtures of regressions are utilized to build sophisticated mappings from high-dimensional data

(hyperspectral images) to low-dimensional data (physical parameters). In Section 4.4, we will provide two additional applications. In the analysis of the orange juice dataset, one uses the high-dimensional spectra to predict the level of sucrose in the juice. In the analysis of the magnetic resonance vascular fingerprint (MRvF) dataset, the goal is to efficiently assess microvascular properties, such as blood volume fraction, vessel diameter, and blood oxygenation in the brain using these MRvF data. All these applications share a common characteristic: predicting a low-dimensional vector variable of interest using high-dimensional features.

Under the high-dimensional setting, data analysis requires special attention. For example, estimating a covariance matrix of high dimension is difficult since the covariance matrix will be singular as the sample size is smaller than the dimension. A significant amount of effort has been made to mitigate the impact of the curse of high dimensionality. Principal component analysis (PCA) is a commonly used practice to reduce dimensions. One disadvantage of using PCA is that the outcomes are difficult to interpret. Furthermore, it is shown that projecting from data on the first few principle directions may not effectively recover the cluster structure (Chang, 1983; McLachlan and Peel, 2000). Another approach is to adopt parsimonious models. In (Banfield and Raftery, 1993), covariance matrices are decomposed into different components. With different constraints on these components, the required number of parameters can be reduced and one can obtain various kinds of parsimonious models. Low-rank approximation such as factor analysis (McLachlan et al., 2003) provides an additional alternative to decompose high-dimensional covariance matrices. Examples include the factor regression model (Bernardo et al., 2003), the cluster-weighted factor analyzers (CWFA) model (Subedi et al., 2013), mixtures of common factor analyzers (Baek et al., 2010) and Gaussian Locally-Linear Mappings (GLLiM) (Deleforge et al., 2015).

GLLiM, albeit being a flexible and powerful tool to predict responses using high-

dimensional features, is known to be sensitive to abnormal data. In mixture modeling, parameters are often estimated by maximum likelihood estimation (MLE) through Expectation-Maximization (EM) algorithms. However, it is well known that MLE is sensitive to outliers. Trimmed likelihood estimators (Cuesta-Albertos et al., 2008; Neykov et al., 2007) are one solution to mitigate the influence of outliers. Data that are unlikely to fit the assumed model are trimmed to eliminate their influence over the estimation. In Markatou (2000), the authors utilized weighted likelihood where the weights are determined by a function that downweighs data points with large residuals. Also see Hadi and Luceño (1997), Müller and Neykov (2003) and Vandev and Neykov (1998) for more references. García-Escudero et al. (2008) used a spurious-outliers model (Gallegos and Ritter, 2005), which decomposes the likelihood into two parts (normal and abnormal data), to handle outliers. It is shown that with a mild assumption on the likelihood function of abnormal data, maximizing such a likelihood function on the full dataset is equivalent to maximizing the likelihood of the normal data. Thus, the outlier removal step can be easily incorporated into the EM procedure. Another family mitigates the impact of outliers by adopting robust estimators (Bai et al., 2012) or utilizing robust distributions such as t- (Chamroukhi, 2016; Perthame et al., 2018) or Laplace distributions (Song et al., 2014).

Another factor that impacts robustness is model stability. The global MLE fails to exist because of the unboundedness of the likelihood function (Day, 1969). When there is more than one cluster, we can construct a cluster centered at a non-zero point and the likelihood will approach infinity as the variance of the mixture (component) decreases to zero. This is also referred to as the "singularity issue" since the problem occurs at several singular points. A common solution is to adopt penalized likelihood (Ciuperca et al., 2003). By penalizing small clusters, we can avoid solutions that lead to infinite likelihood. Furthermore, by carefully selecting the penalized function, the maximum penalized likelihood estimator is found to be strongly consistent (Chen

and Tan, 2009). In the context of Bayesian analysis, the role of the penalized term is replaced by a prior distribution, which shares a similar idea of avoiding small clusters (Fraley and Raftery, 2007). An alternative strategy to tackle the unboundedness of likelihood is to restrict the relative size between clusters. This can be done by adopting a constraint on the relative values on variances (Hathaway, 1985), covariance determinants (McLachlan and Peel, 2004) or eigenvalues of the covariance matrices (García-Escudero et al., 2008).

In this work, we propose a robust Gaussian mixture regression model called RGLLiM (RGLLiM), which builds upon a Gaussian mixture framework for finding high-dimensional non-linear mappings. Our approach focuses on improving model robustness by trimming outliers and by applying a constraint on the relative values of cluster-level eigenvalues. For the latter, we concentrate on building a stable inverse-regression-based prediction. An Expectation-Maximization algorithm is devised to estimate model parameters. A simulation and an experiment on real datasets show that the model performance can be improved with regard to the robustness concern.

4.1 Model specification

There are three specific sub-components that are crucial to the setup of RGLLiM. We will start with the original structure under the regular GLLiM.

Gaussian mixture regressions

The goal is to predict low-dimensional data $Y \in \mathbb{R}^L$ using high-dimensional data $X \in \mathbb{R}^D$, where $D >> L$. Let Z denote the latent variable indicating the cluster assignment and assume that there are K mixtures in total; the joint distribution of

X, Y under the GLLiM framework is expressed in the inverse regression setup:

$$p(X = x, Y = y; \theta) = \sum_{k=1}^{K} p(X = x|Y = y, Z = k; \theta)p(Y = y|Z = k; \theta)p(Z = k; \theta) \tag{4.1}$$

where $\theta = \{c_k, \Gamma_k, \pi_k, A_k, b_k, \Sigma_k\}_{k=1}^{K}$ is the collection of the inverse model parameters. Equation (4.1) describes the inverse relationship mapping from X to Y. The hierarchical structure is defined by:

$$p(X = x|Y = y, Z = k; \theta) = \mathcal{N}(x; A_k y + b_k, \Sigma_k). \tag{4.2}$$

$$p(Y = y|Z = k; \theta) = \mathcal{N}(y; c_k, \Gamma_k), \tag{4.3}$$

$$p(Z = k; \theta) = \pi_k.$$

Nevertheless, the goal here is to predict the low-dimensional X with the high-dimensional Y. This can be achieved by considering the corresponding forward model,

$$p(X = x, Y = y; \theta) = \sum_{k=1}^{K} p(Y = y|X = x, Z = k; \theta^*)p(X = x|Z = k; \theta^*)p(Z = k; \theta^*), \tag{4.4}$$

where

$$\theta^* = \{c_k^*, \Gamma_k^*, \pi_k^*, A_k^*, b_k^*, \Sigma_k^*\}_{k=1}^{K}$$

consists of the forward regression parameters. The relationship between θ and θ^* is

described as follows:

$$\begin{aligned} c_k^* &= A_k c_k + b_k, \\ \Gamma_k^* &= \Sigma_k + A_k \Gamma_k A_k^\top, \\ \pi_k^* &= \pi_k, \\ A_k^* &= \Sigma_k^* A_k^\top \Sigma_k^{-1}, \\ b_k^* &= \Sigma_k^* (\Gamma_k^{-1} c_k - A_k^\top \Sigma_k^{-1} b_k), \\ \Sigma_k^* &= (\Gamma_k^{-1} + A_k^\top \Sigma_k^{-1} A_k)^{-1}. \end{aligned} \tag{4.5}$$

In addition, one further considers the corresponding hierarchical structure:

$$p(Y = y | X = x, Z = k; \theta^*) = \mathcal{N}(y; A_k^* x + b_k^*, \Sigma_k^*). \tag{4.6}$$

$$p(X = x | Z = k; \theta^*) = \mathcal{N}(x; c_k^*, \Gamma_k^*), \tag{4.7}$$

$$p(Z = k; \theta^*) = \pi_k^*.$$

Using the K-mixtures, the prediction of Y is done by taking the expectation over the forward conditional density:

$$\mathbb{E}[Y | X = x; \theta^*] = \sum_{k=1}^{K} \frac{\pi_k^* N(x; c_k^*, \Gamma_k^*)}{\sum_{j=1}^{K} \pi_j^* N(x; c_j^*, \Gamma_j^*)} (A_k^* x + b_k^*), \tag{4.8}$$

in which θ is estimated in the inverse regression, while the prediction is conducted using θ^* as functions of the estimated θ. Recall that the GLLiM procedure eases the estimation task by considering the inverse regression, in which the dimension of the predictors is kept low; and as such, GLLiM avoids the challenging aspects of finding the regression coefficients for a set of high-dimensional predictors.

Under the GLLiM framework, X can be partially latent. That is, X can be

decomposed as observed component T and latent component W. That is,

$$X = \begin{bmatrix} T \\ W \end{bmatrix}, \tag{4.9}$$

where $T \in \mathbb{R}^{L_t}$, $W \in \mathbb{R}^{L_w}$ and $L_w + L_t = L$. Assuming that T and W are independent given Z, we have

$$c_k = \begin{bmatrix} c_k^t \\ c_k^w \end{bmatrix}, \Gamma_k = \begin{bmatrix} \Gamma_k^t & 0 \\ 0 & \Gamma_k^w \end{bmatrix} \text{ and } A_k = \begin{bmatrix} A_k^t, A_k^w \end{bmatrix}. \tag{4.10}$$

Trimmed likelihood

Let $\{x_n, t_n\}_{n=1}^N$ be the realization for X and T. Let $P = P_{(T,X)}$ be the probability measure induced by the joint distribution of X and T. GLLiM uses cluster analysis based on the inverse regression setup in (4.1), with Y replaced by T to obtain estimated θ. Let $D(T, X; \theta) = p(T = t, X = x; \theta)$. The corresponding trimmed likelihood setup in GLLiM is to identify $\hat{\theta}$ that maximizes

$$\sum_{n=1}^{N} \left[\log\{D(t_n, x_n; \theta)\} I_{R_o^C(\theta)}(t_n, x_n) \right], \tag{4.11}$$

where $I_A(\cdot)$ denotes the indicator function of set A, and data within the partition R_0 are considered outliers. Specifically, one uses the indicator function to keep all data points within the complement of $R_0(\theta)$, namely $R_0^c(\theta)$, in the trimmed log-likelihood. At the population level, we denote

$$L(\theta, P) = E_P \left[\log\{D(T, X; \theta)\} I_{R_o^C(\theta)}(T, X) \right]. \tag{4.12}$$

Conceptually, the goal is to construct an estimator of θ belonging to a pre-determined domain that, at the population level, maximizes $L(\theta, P)$ subject to $P(I_{R_\alpha^C(\theta)}(T, X)) \geq 1 - \alpha$. García-Escudero et al. (2008) proposed to create an additional cluster to contain those data points to be trimmed off and a corresponding procedure to carry out the outlier classification. We adopt the identical procedure under the inverse regression setting.

The eigenvalue ratio constraint

For the stability concern, we aim to put constraints on covariance matrices and restrict the parameter space for θ, or equivalently θ^*. This type of restriction is an extension of those introduced by Hathaway (1985) for scalar data. The procedure allows avoiding the singularities introduced by potentially very different covariance matrices by controlling the ratio between the maximum and the minimum eigenvalues of these matrices. Following the mixture regression structure in García-Escudero et al. (2017), such constraints should be imposed on Σ_k's and Γ_k's. Here, we consider a different direction by putting Eigenvalue Ratio (ER) constraints on Γ_k^* given in Equation (4.5). Denote $\lambda_d(\cdot)$ as the d-th eigenvalue of the input matrix. Specifically, the ER constraint requires

$$\frac{\max_k \max_d \lambda_d(\Gamma_k^*)}{\min_k \min_d \lambda_d(\Gamma_k^*)} \leq C, \tag{4.13}$$

where $C \geq 1$ is a fixed constant. That is, the relative size of Γ_k^* for all k is controlled by restricting the ratio between the maximum and minimum eigenvalues. For simplicity, we use Γ_k^* to denote the covariance matrix fulfilling the ER constraint. If the original Γ_k^* does not fulfill the requirement, we update Γ_k^* using the method described in the

later section and the corresponding forward parameters are updated accordingly by

$$\pi_k^* = \pi_k$$

$$c_k^* = A_k c_k + b_k$$

$$\Sigma_k^* = \Gamma_k - \Gamma_k A_k^\top (\Gamma_k^*)^{-1} A_k \Gamma_k \tag{4.14}$$

$$A_k^* = \Gamma_k A_k^\top (\Gamma_k^*)^{-1} \tag{4.15}$$

$$b_k^* = c_k - \Gamma_k A_k^\top (\Gamma_k^*)^{-1} (A_k c_k + b_k) \tag{4.16}$$

Finally the prediction is done by $\mathbb{E}[Y|X = x; \theta^*]$ using Equation (4.8).

4.2 Expectation-Maximization algorithm

Given the training dataset $\{x_n, t_n\}_{n=1}^N$, the latent component $\{w_n\}_{n=1}^N$ and the cluster-indicator variable $\{Z_n\}_{n=1}^N$, we can estimate the parameters using the Expectation-Maximization algorithm under the ER constraint. Note that without the ER constraint, the complete data likelihoods, as if Z were observed, parametrized using θ and θ^* can be considered to be equivalent. To overcome the high dimensionality issue when estimating large covariance matrices, we first consider the inverse regression setting in Equation (4.1), estimate the inverse parameters, θ, and check the validity of the ER constraint. If the ER constraint is satisfied, we directly obtain θ^* from θ and carry out the prediction. On the other hand, if the ER constraint is not fulfilled, we then convert the inverse parameters to the forward parameters and update Γ_k^*.

The target function based on the inverse parameters is now updated from Equation (4.12) to

$$\mathbb{E}_P[\log p((X, T, W; \theta) I_{\mathcal{A}(\theta)}(X, T)], \tag{4.17}$$

accounting for the latent W and maximizing over all possible sets of $\mathcal{A}(\theta)$ for the

optimal set of $R_0^c(\theta)$.

The EM algorithm contains a maximization step to estimate parameters and two expectation steps for estimating, two posterior distributions, $p(Z|(y,t);\theta)$ and $p(W|(y,t,Z);\theta)$.

E-W step

The E-W step that aims to estimate the distribution $p(w_n|Z_n = k, t_n, y_n; \theta)$ is Gaussian with mean μ_{nk}^w and covariance matrix S_k^w where:

$$\begin{aligned}\mu_{nk}^w &= S_k^w \left(A_k^{w\top} \Sigma_k^{-1} (y_n - A_k^t t_n - b_k) + (\Gamma_k^w)^{-1} c_k^w \right), \\ S_k^w &= \left((\Gamma_k^w)^{-1} + A_k^w \Sigma_k^{-1} A_k^w \right)^{-1}\end{aligned}$$

E-Z step

The original EM algorithm presented in Deleforge et al. (2015) only depends on the inverse parameters. However, the estimation of the cluster assignment posterior probability r_{nk} should depend on $\tilde{\Gamma}_k^*$ if the ER constraint is not satisfied. The posterior probability of cluster assignment is given by

$$r_{nk} = p(Z_n = k|y_n, t_n; \theta^*) = \frac{\pi_k^* p(y_n, t_n|Z_n = k; \theta^*)}{\sum_{j=1}^K \pi_j^* p(y_n, t_n|Z_n = j; \theta^*)},$$

where

$$\begin{aligned}p(y_n, t_n|Z_n = k; \theta^*) &= p(t_n|y_n, Z_n = k; \theta^*) p(y_n|Z_n = k; \theta^*) \\ p(t_n|y_n, Z_n = k; \theta^*) &= \mathcal{N}(t_n; \tilde{\mu}_k, \tilde{\Sigma}_k), \\ \tilde{\mu}_k &= c_k^t + \Gamma_k^t A_k^{t\top} (\Gamma_k^*)^{-1} (y_n - A_k^t c_k^t - A_k^w c_k^w - b_k) \\ \tilde{\Sigma}_k &= \Gamma_k^t - \Gamma_k^t A_k^{t\top} (\Gamma_k^*)^{-1} A_k^t \Gamma_k^t \\ p(y_n|Z_n = k; \theta^*) &= \mathcal{N}(y_n; A_k^t c_k^t + A_k^w c_k^w + b_k, \Gamma_k^*),\end{aligned} \tag{4.18}$$

For more details, please refer to Appendix A.1. We update the posterior probability with

$$r_{nk} = \begin{cases} 1 & \text{if } r_{nk} = \max_k\{r_{n1}, ..., r_{nK}\} \\ 0 & \text{otherwise} \end{cases} \tag{4.19}$$

to form disjoint partitions.

C step

In the concentration step (C Step) data are removed from the current iteration following the procedure given in García-Escudero et al. (2008). This can be done by setting the posterior probability r_{nk} to zero. Consider the discriminant functions defined as

$$\mathcal{L}_k((y_n, t_n); \theta^*) = \pi_k p(T = t_n, Y = y_n | Z = k; \theta^*)$$
$$\mathcal{L}((y_n, t_n); \theta^*) = \max\{\mathcal{L}_1((y_n, t_n); \theta^*), ..., \mathcal{L}_K((y_n, t_n); \theta^*)\}.$$

Given $0 < \alpha < 1$ as a fixed trimming level, we let $\mathcal{L}_\alpha$ denote the α-quantile of $\{\mathcal{L}((y_1, t_1); \theta^*), ..., \mathcal{L}((y_N, t_N); \theta^*)\}$. Considering $n^* \in \{n | \mathcal{L}(\{y_n, x_n\}; \theta) < \mathcal{L}_\alpha\}$, we set $r_{n^*k} = 0$ for all $k = 1, ..., K$.

M step

The M step is targeting on estimating $\theta = \{c_k, \Gamma_k, \pi_k, A_k, b_k, \Sigma_k\}_{k=1}^K$ and applying the ER constraint on Γ_k^*. The first part is the same as described in Deleforge et al. (2015) and thus is omitted here. We focus on putting the ER constraint on Γ_k^*. Maximizing the log-likelihood function against Γ_k^* is equivalent to minimizing:

$$\sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left(\log |\Gamma_k^*| + \text{trace}((\Gamma_k^*)^{-1} S_k) \right), \tag{4.20}$$

where S_k is the high-dimensional sample covariance matrix and can be obtained

using the inverse parameters $S_k = \Sigma_k + A_k \Gamma_k A_k^\top$. Denote $S_k = V_k \Lambda_k V_k^\top$, where $\Lambda_k = \text{diag}(\lambda_{k1}, ..., \lambda_{kD})$ and V_k is the matrix composed by eigenvectors of S_k. Using the results in García-Escudero et al. (2008), minimizing Equation (4.20) is equivalent to minimizing

$$\sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \sum_{d=1}^{D} \left(\log(\lambda_{kd}^m) + \frac{\lambda_{kd}}{\lambda_{kd}^m} \right), \tag{4.21}$$

where $\lambda_{k,d}^m$ is the truncated eigenvalue defined as:

$$\lambda_{kd}^m = \begin{cases} d_{kd} & \text{if } \lambda_{kd} \in [m, Cm] \\ m & \text{if } \lambda_{kd} < m \\ Cm & \text{if } \lambda_{kd} > Cm \end{cases} \tag{4.22}$$

The high-dimensional covariance matrix is updated by

$$\Gamma_k^* = V_k \Lambda_k^m V_k^\top, \tag{4.23}$$

where $\Lambda_k^m = \text{diag}(\lambda_{k,1}^m, ..., \lambda_{k,D}^m)$ is the matrix with truncated eigenvalues on the diagonal. At the forward regression setting, to avoid singularity, it is also necessary to add an extra constraint so that $\tilde{\Sigma}_k$ for all k is a valid covariance matrix. This requirement is carried out by finding the appropriate m. Details can be found in Appendix A.2.

4.3 Simulation studies

4.3.1 Simulation settings

We simulated datasets with different cluster structures using the orange juice dataset. Recall that the orange juice dataset contains the spectra (Y) and the corresponding level of sucrose (T) measured on different kinds of orange juice. In addition, the dataset contains several abnormal data that could impair the model performance.

We follow the procedure in Perthame et al. (2018) to make $D \approx N$. There are 218 samples and, the dimension of each spectrum is 134 after the pre-processing procedure. Details of identifying cluster information and simulation procedure can be found in Appendix A.3.

Using the procedure described in Appendix A.3, we identify four groups. Figure 4.1 shows an illustration of these groups. Groups 1, 2 and 4 are distinct subsets, while Groups 2 and 3 overlap. We adopt a procedure similar to that described in Section 2.4.2, and identify 11 outliers which will be used to evaluate the robustness of different methods. We simulate the data under four cases: distinct clusters without outliers, distinct clusters with outliers, overlapped clusters without outliers and overlapped clusters with outliers. Hereafter, we will refer to them as Case 1 to Case 4. In each simulation, we chose 3 clusters: we simulate data from Groups 1, 2 and 4 for the distinct-cluster scenarios (Cases 1 and 2), and from Groups 1, 2 and 3 for the overlapped-cluster scenarios (Cases 3 and 4), respectively. We generate 200 data points for each cluster: 100 for the training dataset, and 100 for the testing dataset. Consequently, each simulation consists of 300 normal training data and 300 testing data. For the cases with outliers, there are 11 additional abnormal data, and thus, the sizes of the training datasets for Cases 2 and 4 are 311.

We first compare the performance of different methods when the tuning parameters are ideally determined. That is, for all methods, we set $K = 3$, $L_w = 8$. Furthermore, we consider three different settings of RGLLiM, RGLLiM 1 to RGLLiM 3, with $(C = 10^5, \alpha = 0)$, $(C = 10^5, \alpha = 0.05)$ and $(C = \infty, \alpha = 0.05)$, respectively. In addition, we also compare the performance to that of SLLiM (Perthame et al., 2018), which is an alternative approach to "robustify" the regular GLLiM.

The numerical properties of the methods are evaluated using the prediction mean squared errors (PMSE), clustering accuracy and the Rand index (Rand, 1971). The PMSE is calculated as $\frac{1}{N}\sum_{n=1}^{N} ||\hat{t}_n - t||_2^2$, where N is the total number of the data;

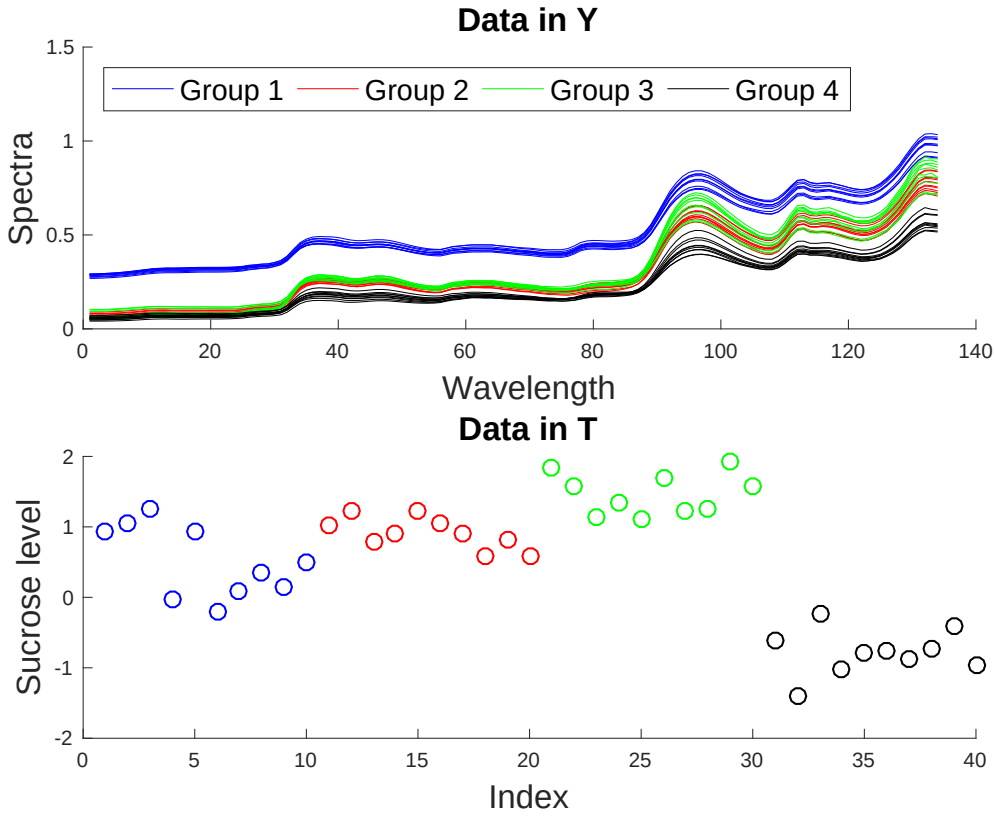


Figure 4.1: A subset of simulated data from Groups 1–4. The upper and the lower panels show the high-dimensional Y and the low-dimensional T, respectively.

t_n, $\hat{t}_n$ are the n-th true and predicted values, respectively. We are also interested in clustering accuracy. The associations between Y and T differ across different clusters. Thus, the clustering results are important to establish the estimated model. For clustering, the labels may switch between two clusters; thus we have to find the mapping between the true clusters and the estimated ones. We establish the mapping based on the majority of the data coming from each true cluster. As an example, if most of the data points coming from true cluster 1 are assigned to estimated cluster 2, we will treat true cluster 1 and estimated cluster 2 as the same cluster. Every sample coming from true cluster 1 that is assigned to estimated cluster 2 is regarded as achieving a correct assignment. However, the clustering accuracy may not be informative when two clusters merge together. Thus, we further use the Rand index

to assess the quality of the cluster assignment. In addition, we provide the PMSEs for the data points that are clustered correctly (*Corr. cluster*) and those that are assigned erroneously (*Mis. cluster*).

Criteria	GLLiM	RGLLiM 1	RGLLiM 2	RGLLiM 3	SLLiM
			Training		
PMSE	1.102e-08	1.102e-08	1.195e-08	1.059e-08	6.314e-09
(Corr. class)	1.102e-08	1.102e-08	1.195e-08	1.059e-08	6.314e-09
(Miss class)	-	-	-	-	-
Cluster accuracy	100.00	100.00	100.00	100.00	100.00
Rand index	1.0000	1.0000	1.0000	1.0000	1.0000
			Testing		
PMSE	1.103e-08	1.103e-08	1.316e-08	1.209e-08	6.313e-09
(Corr. class)	1.103e-08	1.103e-08	1.316e-08	1.209e-08	6.313e-09
(Miss class)	-	-	-	-	-
Cluster accuracy	100.00	100.00	100.00	100.00	100.00
Rand index	1.0000	1.0000	1.0000	1.0000	1.0000

Table 4.1: The training and testing performances under Case 1 using the known parameters.

Criteria	GLLiM	RGLLiM 1	RGLLiM 2	RGLLiM 3	SLLiM
			Training		
PMSE	0.007529	0.006892	9.867e-09	3.947e-05	0.001918
(Corr. class)	0.007529	0.006892	9.867e-09	3.934e-05	0.00192
(Miss class)	-	-	-	-	1.071e-05
Cluster accuracy	100.00	100.00	100.00	100.00	99.90
Rand index	0.9777	1.0000	1.0000	1.0000	0.9322
			Testing		
PMSE	0.007529	0.006892	1.06e-08	6.307e-05	0.001918
(Corr. class)	0.007529	0.006892	1.06e-08	6.307e-05	0.001918
(Miss class)	-	-	-	-	-
Cluster accuracy	100.00	100.00	100.00	100.00	100.00
Rand index	0.9777	1.0000	1.0000	1.0000	0.9331

Table 4.2: As in Table 4.1, under Case 2.

Table 4.1 to Table 4.4 show the performance of different methods under the four different cases. Case 1 is the simplest scenario, and all methods can recover the cluster assignment perfectly. RGLLiM will add some biases when $C < \infty$; thus we see that the performance of RGLLiM 1 and RGLLiM 2 is slightly worse than that of other methods. However, the prediction performance is still satisfactory. In Case 2, the training dataset contains several outliers. Even though all methods can recover the

Criteria	GLLiM	RGLLiM 1	RGLLiM 2	RGLLiM 3	SLLiM
			Training		
PMSE	0.01566	0.01248	0.01216	0.01188	0.01753
(Corr. class)	0.01478	0.01291	0.01284	0.01213	0.01835
(Miss class)	0.02476	0.00919	0.005545	0.009198	0.000126
Cluster accuracy	91.23	88.43	90.67	91.47	95.53
Rand index	0.7771	0.8128	0.8069	0.8090	0.6608
			Testing		
PMSE	0.02866	0.02609	0.02676	0.02667	0.02737
(Corr. class)	0.02872	0.02591	0.02605	0.02651	0.02737
(Miss class)	0.02454	0.02993	0.04577	0.03656	-
Cluster accuracy	98.47	95.43	96.37	98.43	100.00
Rand index	0.8104	0.8445	0.8147	0.8266	0.6433

Table 4.3: As in Table 4.1, under Case 3.

Criteria	GLLiM	RGLLiM 1	RGLLiM 2	RGLLiM 3	SLLiM
			Training		
PMSE	0.02515	0.02536	0.02219	0.02088	0.0286
(Corr. class)	0.02466	0.02504	0.02195	0.02126	0.02958
(Miss class)	0.04788	0.02816	0.02373	0.01799	4.971e-09
Cluster accuracy	97.90	89.50	87.63	87.99	96.70
Rand index	0.7653	0.7910	0.8058	0.7977	0.6959
			Testing		
PMSE	0.02798	0.02905	0.02426	0.02708	0.03056
(Corr. class)	0.02783	0.029	0.02428	0.02683	0.03056
(Miss class)	0.09181	0.03022	0.02407	0.0331	-
Cluster accuracy	99.77	95.77	92.67	96.00	100.00
Rand index	0.7755	0.8029	0.8140	0.8076	0.6878

Table 4.4: As in Table 4.1, under Case 4.

correct cluster assignment perfectly, we observe that the prediction errors increase when the outlier removal technique is not adopted (GLLiM and RGLLiM 1). The presence of outliers can distort the parameter estimation, which impairs the prediction performance. This shows the necessity of outlier removal. On the other hand, SLLiM mitigates the impact of outliers by assuming t-distributed clusters. We see that the performance of SLLiM is better than that of GLLiM and RGLLiM 1, but is still worse compared to RGLLiM 2 and RGLLiM 3. Case 3 is the scenario in which

two clusters overlap. Under this case, all methods perform similarly except SLLiM. The fact of high clustering accuracy but low Rand index shows that overlapped clusters are merged together. Suppose we have data points from cluster 1 and 2 being mapped to the identical estimated training cluster. When calculating clustering accuracy, as long as the data points coming from one true cluster are assigned to the same estimated cluster, the cluster assignment is regarded as correctly obtained. However, the overall estimated cluster assignments are pretty different from the original true assignment, which causes the low Rand index. For Case 4, a similar phenomenon happens to GLLiM. Because of the outliers, it is more likely to form large clusters, and these large clusters will absorb other clusters through EM iterations. With the constraint of relative cluster size, we obtain rather balanced clustering results using RGLLiM. We also reach a higher Rand index using RGLLiM, which indicates that the original clustering assignments are better recovered when the ER constraint is adopted, compared to other methods. As for RGLLiM 3, even though no ER constraint is applied, by trimming outliers during the estimation process, it results in a similar outcome.

4.3.2 Parameter selection

The parameters can be selected using cross-validation. The first task is to determine the range of each tuning parameter. For parameters (L_w, C, α), the range is easy to determine. The parameter L_w reflects the latent structure and, based on what is reported in other publications and our own experiences, a number smaller than 20 should be sufficient. The upper bound of the constant C can be found by simply performing RGLLiM with a very large C and calculating the eigenvalue ratio between the estimated Γ_k^*. This practice will provide a useful upper bound of C. The value of α is supposed to be small. By evaluating the data points being trimmed, we can determine whether a larger than necessary α has been chosen. Determining the

maximum value of K, on the other hand, is a challenging task since its value could vary widely, so as to reflect different data intrinsic structures. In this section, we propose a method to find the upper bound of K when performing cross-validation.

To determine the upper bound of K, we select $\alpha > 0$ so as to eliminate the impact of outliers. The value of α should be small. In addition, there is an interactive trend between K and L_w. For a large K, we would obtain simple within-cluster structures, and thus the value of L_w tends to be small. For a smaller K, the cluster structure would be more complicated and thus one might need a larger L_w to capture the latent structure. In order to find the upper bound of K, we could choose a small L_w, e.g., $L_w = 0$ or $L_w = 2$. As for C, due to the nature of the constraints, we observe that a less restrictive constraint would lead to selecting a smaller K. Thus, we could pick an arbitrary large C. We next calculate the relative increment ratio of the log-likelihood. The relative increment ratio is calculated as the difference between two consecutive log-likelihoods of K and $K + 1$ divided by the log-likelihood of $K + 1$.

Figure 4.2 shows the log-likelihood and the log-likelihood relative increment ratio under different cases. For each case, we set $C = 10^7$, $\alpha = 0.05$. We choose 1% as the threshold and pick the K for which the corresponding log-likelihood relative increment ratio first drops below 1% as the upper bound. From Figure 4.2, if $L_w = 0$ is considered, selecting the upper bound of K to be 15 seems to be sufficient. If we use $L_w = 2$, the upper bound of K would be 8 for Cases 1, 3 and 4. For Case 2, we could select 5 as the upper bound of K. We enlarge these selected upper bounds to 20 and further pick $K = 30, 50$ to evaluate the performance of using a larger-than-needed K.

We next compare the results when the tuning parameters are determined using the reported model selection techniques. For GLLiM and SLLiM, the tuning parameters are selected using BIC. We use the method reported in this section to select the tuning parameters for RGLLiM. We choose K from values of 2 to 10, 30, 50; L_w from values of 5 to 20; C from 10^5, 10^7, ∞ and α from 0, 0.05, 0.1. The parameters selected

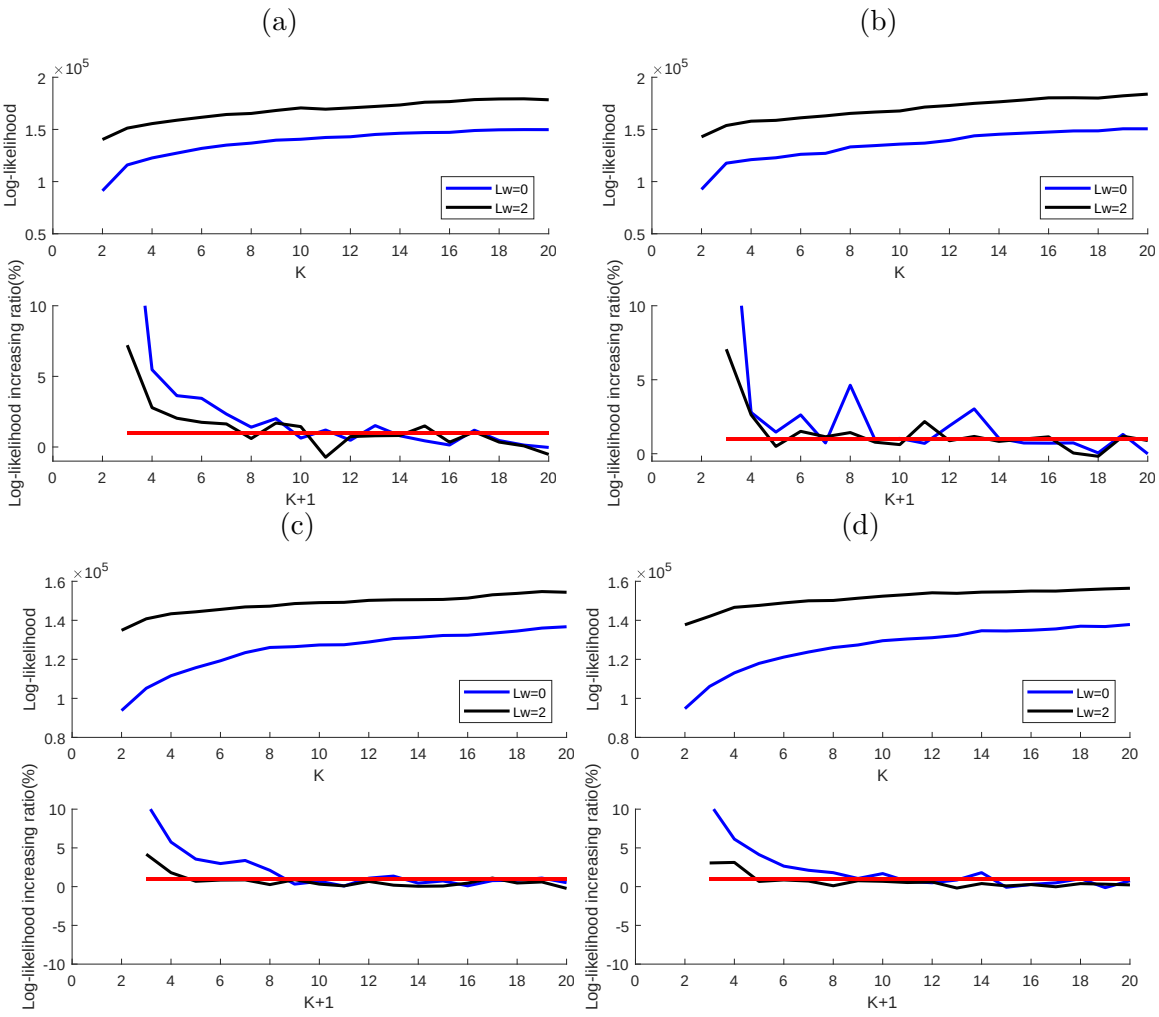


Figure 4.2: Changes of log-likelihood and the relative increment ratio for $C = 10^7$, $\alpha = 0.05$. Within plots (a)–(d), the upper panel shows the log-likelihood values for different K, and the lower panel shows the log-likelihood relative increment ratios, changing with $K + 1$. Plots (a)–(d) correspond to (a) Case 1; (b) Case 2; (c) Case 3 and (d) Case 4.

under each case are shown in Table 4.5.

	GLLiM	RGLLiM	SLLiM
Case 1	$K = 3, L_w = 10$	$K = 3, L_w = 10, C = 10^7, \alpha = 0$	$K = 3, L_w = 8$
Case 2	$K = 3, L_w = 11$	$K = 3, L_w = 12, C = 10^7, \alpha = 0.05$	$K = 3, L_w = 10$
Case 3	$K = 3, L_w = 6$	$K = 3, L_w = 11, C = 10^7, \alpha = 0$	$K = 3, L_w = 6$
Case 4	$K = 3, L_w = 12$	$K = 3, L_w = 8, C = 10^5, \alpha = 0.05$	$K = 3, L_w = 6$

Table 4.5: The parameter settings under different cases.

Table 4.6 shows the prediction performance under different cases. The prediction

MSEs for GLLiM on Case 2 and Case 4 are larger than those for RGLLiM and SLLiM because of the outliers. For RGLLiM, the overall performance is similar to what we have obtained when the parameter settings are ideally chosen. As for SLLiM, it selects parameters that are close to the true ones and thus reaches a similar performance.

	GLLiM	RGLLiM	SLLiM
		Training	
Case 1	1.102e-08	1.102e-08	6.314e-09
Case 2	0.0003334	7.863e-07	3.347e-08
Case 3	0.01557	0.01328	0.01802
Case 4	0.0228	0.02219	0.03584
		Testing	
Case 1	1.103e-08	1.103e-08	6.313e-09
Case 2	0.0003334	8.341e-09	3.374e-08
Case 3	0.02803	0.02521	0.02568
Case 4	0.03027	0.02426	0.03480

Table 4.6: Comparison of the performance of different methods under different cases when parameters are selected using the reported model selection strategies.

4.4 Numerical investigation using real-world datasets

4.4.1 The orange juice data

The orange juice data is used in the simulation study in the previous section. In this section we evaluate the performance of different methods when applied to the dataset itself. We ran 20 experiments. For each experiment, the training and testing datasets were selected as follows. There are 218 samples in the orange juice dataset. We randomly selected 20 data points from a subset of the data, excluding outliers, as the testing samples. The rest of the 198 data points, including outliers, are treated as the training samples. This setup enables us to clearly assess the impact of outliers in each training dataset, without letting deteriorated performances in predicting outliers in the testing samples mask the quality of the prediction of regular data points. We compared the performance of several methods, including GLLiM, SLLiM and RGLLiM. All methods were randomly initialized 10 times and the result with the largest likelihood was selected. We followed the tuning procedures described in Perthame et al. (2018) for GLLiM and SLLiM. We fixed $K = 10$ and selected L_w using BIC for GLLiM and SLLiM. For RGLLiM, the tuning parameters were selected through cross-validation.

	GLLiM	SLLiM	RGLLiM
PMSE	0.4360	0.3337	0.1287

Table 4.7: The prediction performance using different methods.

Table 4.7 shows the average prediction MSE over 20 experiments. With no robustness control, GLLiM behaves unsurprisingly the worst among the three methods. SLLiM mitigates the influence of outliers through having t-distributed clusters, but it does not contain any mechanism for controlling the relative cluster sizes. RGLLiM controls the relative cluster sizes to prevent forming clusters which are either too small or too large. By mitigating the impact of outliers on clustering, RGLLiM achieves

better prediction performance.

We next analyze the outliers identified by using different choices of C. We denote RGLLiM(10^5) and RGLLiM(∞) as the RGLLiM approaches with $C = 10^5$ and $C = \infty$, respectively. The rest of the tuning parameters are kept the same: $K = 10, L_w = 10$ and $\alpha = 0.05$. The top 5 outliers identified by these two settings are shown in Table 4.8. The only common outlier identified by both settings is data point 133. This indicates that the choices of C can influence the identification of outliers. To evaluate which choice provides a better selection of outliers, we calculate the prediction MSE as follows. By setting $\alpha = 0.05$, RGLLiM will remove 10 data points. We let A be the set of the data points removed by RGLLiM(10^5), and B, by RGLLiM(∞). We calculate the prediction MSE of $B \setminus A$ using the RGLLiM(10^5) model and vice versa, where the set $B \setminus A$ contains the outliers identified by RGLLiM(∞) but not by RGLLiM(10^5). That is, we are actually calculating the in-sample prediction MSE of RGLLiM(10^5). Similarly, we can obtain the in-sample prediction MSE for set $A \setminus B$ of RGLLiM(∞). The in-sample prediction MSEs of RGLLiM(10^5) and RGLLiM(∞) are 0.0104 and 0.0147, respectively. A smaller in-sample prediction MSE indicates that the data points fit better with other training data points. Thus, RGLLiM(10^5) provides a slightly better selection of outliers. The top 5 outliers identified by RGLLiM(10^5) are shown in Figure 4.3, and we note that the data points with recognizable abnormal patterns are identified.

	Top 5 outliers
RGLLiM $C = 10^5$	37, 42, 130, 133, 194
RGLLiM $C = \infty$	133, 137, 140, 150, 192

Table 4.8: The top 5 data being removed.

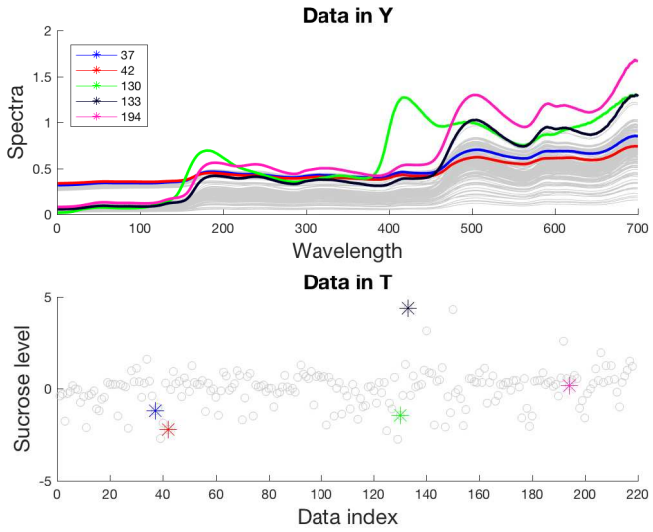


Figure 4.3: The orange juice dataset. The upper panel shows the high-dimensional data (Y) and the lower panel shows the low-dimensional data (T).

4.4.2 The fingerprint data

The fingerprint dataset is introduced and fully analyzed in Chapter III. As a reminder, we are interested in predicting the microvascular properties mean vessel radius (Radius) and Blood Volume Fraction (BVf), and the measurement of blood oxygenation (DeltaChi) using the fingerprints. As shown in Figure 4.4, the dictionary contains several measurements of the fingerprint with extreme values and this could result in robustness concerns during the model-building exercise. We first analyze a subset with potential outliers and demonstrate the necessity of robustness control. Next, we evaluate the performance of the dictionary matching method (Lemasson et al., 2016), GLLiM and RGLLiM using the synthetic dataset. Finally, the results on the data acquired from in vivo experiments are presented.

4.4.2.1 Subset with abnormal data

As the analysis described in Section 3.2.1, we separate the full synthesized dataset into 20 groups. To demonstrate the advantage of the proposed method for robustness, we first select a complex subset with outliers. Visually notable outliers mostly

appear in Group 3 to Group 8, with apparently high peaks in the high-dimensional X (fingerprint). The 98% percentile of the peak values is 26.3612. Using this as the decision threshold, we identify 244 outliers out of 12180 data. Since the Radius (t_1) is fixed in Groups 3 to 8, we focus on predicting BV (t_2) and DeltaChi (t_4). We randomly select 5000 normal data and 200 abnormal data to form a training dataset. We show this selected subset with outliers in Figure 4.4.

For method evaluation, we select 1000 testing samples from the normal data. The tuning parameters are selected through cross-validation. For GLLiM, we select $K = 200, L_w = 20$. The resulting testing MSE is 0.002151. For RGLLiM, we select $K = 150, L_w = 20, C = 10^{10}, \alpha = 0.05$, and obtain the testing MSE of value 0.001228, a 43% reduction of that obtained by GLLiM.

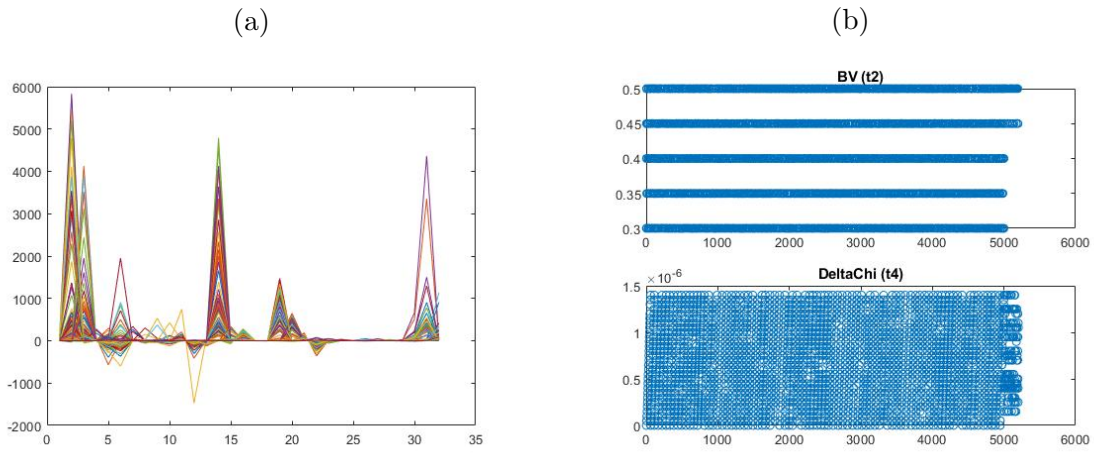


Figure 4.4: The fingerprint subset with outliers: (a) data in high dimension (Y) and (b) data in low dimension (T).

4.4.2.2 Performance on the synthetic data

The synthetic dataset is separated into 20 folds. Table 4.9 shows the results of the dictionary matching, GLLiM and RGLLiM. The numbers of training and testing data for each group are described in Section 3.2.1. For RGLLiM, the cross-validation suggests the choice of $K = 100, L_w = 20, C = 10^{10}, \alpha = 0.02$. For GLLiM, we provide

results with $K = 100$ and $L_w = 20$, as a comparison. The models estimated from each group are combined together into the final model used for prediction. Table 4.9 shows the 50%, 90% and 99% quantiles of the squared prediction errors, which indicates the "average", "almost-all" and the "worst" performances of different methods. BVf is the microvascular parameter of the most interest, and RGLLiM outperforms the other two methods. As for Radius, RGLLiM is slightly worse than GLLiM for the 90% quantile, but its performance is better on the median. As for DeltaChi, RGLLiM performs better than GLLiM on the 90% quantile but worse than GLLiM on the 99% quantile. This is because RGLLiM trims off outliers to reach robustness in estimation, and this practice may lead to worse performance for certain extreme cases.

	Dictionary matching			GLLiM			RGLLiM		
	50%	90%	99%	50%	90%	99%	50%	90%	99%
Radius	0	0.2843	21.44	1.2189×10^{-3}	0.2074	21.44	1.1075×10^{-3}	0.2141	21.44
BVf	0	0	0.0023	8.113×10^{-15}	1.744×10^{-8}	2.267×10^{-4}	1.08×10^{-27}	3.676×10^{-8}	1.398×10^{-4}
DeltaChi	0	0.0143	0.3571	2.548×10^{-10}	4.125×10^{-7}	0.01646	2.385×10^{-10}	3.387×10^{-7}	0.03113

Table 4.9: The quantiles of the fingerprint data. Note that the dictionary matching method adopts the microvacular properties from the nearest training as the predicted values. It is possible that the predicted squared error equals zero since the synthetic data is generated on a grid.

4.4.2.3 Performance on the in vivo fingerprint data

In order to evaluate the prediction quality on the in vivo fingerprint data, we first locate the best-matched dictionary data point for each in vivo fingerprint measurement. We include the data within the region of the interest (ROI) whose $r^2 > 0.8$ (Lemasson et al., 2016). There are 88113 fingerprints inside the ROI, corresponding to 44956 unique dictionary fingerprints. Most included dictionary data points belong to Group 1 (96.85%). The second and the third largest groups that contain the matched dictionary fingerprint are Group 2 (1.75%) and Group 18 (1.21%). In addition to the synthetic training data, we also include a small number of in vivo fingerprint data. Since the fingerprints from the animal study are noisier than the synthetic ones, adding the in vivo fingerprints enables the model to accommodate the in vivo samples in prediction. The ratio of the synthetic samples to the in vivo samples is 4 to 1. The cross-validation outcomes suggest the choice of the tuning parameters as $K = 150, L_w = 10, C = 10^{10}, \alpha = 0.05$ for RGLLiM. We compare the results to GLLiM with $K = 150, L_w = 10$. Since there is no ground truth for the Radius, BVf and DeltaChi of the in vivo fingerprints, we compare the prediction results of GLLiM and RGLLiM to the results obtained by the dictionary matching method. Table 4.10 shows the results. We note that for both methods, the predicted values of BVf are close to the ones obtained by the dictionary matching method. As for DeltaChi, RGLLiM outcomes are uniformly closer to the targets than GLLiM outcomes. The same does not hold for Radius, with the 99% quantile of squared errors obtained by RGLLiM being higher than those obtained by GLLiM.

	GLLiM			RGLLiM		
	50%	90%	99%	50%	90%	99%
Radius	2.7029×10^{-4}	0.0060	0.1118	2.0809×10^{-4}	0.0054	0.1760
BVf	2.9514×10^{-6}	4.9759×10^{-5}	7.1762×10^{-4}	1.1152×10^{-6}	3.2134×10^{-5}	5.6165×10^{-4}
DeltaChi	0.0022	0.0349	0.3553	0.0016	0.0338	0.3035

Table 4.10: The quantiles of squared errors for the fingerprint data.

	Dictionary matching			GLLiM			RGLLiM		
	50%	90%	99%	50%	90%	99%	50%	90%	99%
9L	1.1180	3.9803	10.6829	0.4535	2.8226	4.7301	0.2004	2.0813	4.7020
C6	1.1208	4.4719	14.4888	0.9198	5.1085	14.1683	0.7610	6.3971	15.3170
F98	1.0994	4.2373	14.4888	0.9886	5.3409	15.1071	0.7135	4.1853	13.5320
Stroke	1.1663	5.8045	14.8888	1.0368	3.1978	4.7664	0.1270	2.5659	4.7033
Health	1.0931	3.8086	7.7912	1.4402	8.9761	30.9789	0.9876	8.9632	30.7002

Table 4.11: The 50%, 90% and 99% quantiles of ADC squared errors for different methods on different image categories. For each entry, the result for the best performer from the three methods is underlined.

The ground truth of ADC is available, and thus we compare the quality of the predictions of ADC using all three methods in Table 4.11. We observe that both RGLLiM and GLLiM outperform the dictionary matching method and that RGLLiM predicts better than GLLiM, in general, as shown by the median comparisons. A tradeoff of the robustness control of RGLLiM is that the prediction performance of some extreme cases would be slightly worse.

4.5 Conclusion

In this chapter we proposed RGLLiM to combat outliers and to avoid the singularity issue. Outliers are removed by maximizing the trimmed likelihood. The singularity issue is resolved by putting a constraint on the relative cluster size. Results on the simulated and the real orange juice dataset demonstrate that RGLLiM is insensitive to outliers. In addition, with the ER constraint, RGLLiM can better identify abnormal data. Using the fingerprint dataset, we show that RGLLiM is capable of handling datasets with complicated structures. RGLLiM establishes the primary patterns of the data, which may sacrifice the prediction performance in some extreme instances. However, these extreme cases can be easily detected by users using the posterior probabilities as discussed in Chapter III. Moreover, the cluster structure and the outliers identified by RGLLiM can be utilized for further analyses.

CHAPTERS V

Zeroth Order Optimization Method for Adversarial Example Generation

Traditional machine learning algorithms assume no interaction between the data generating process and the model. However, in many real-world applications, this assumption is invalid. As an example, the performance of a spam filtering model can downgrade quickly after it is deployed. Spammers learn to insert non-spam words into email to fool the filter. As the spam filter is updated with these tricks, spammers develop new techniques to bypass the filter (Brückner et al., 2012; Fawcett, 2003). Similar arms races can be found in other fields such as fraud detection (Fawcett and Provost, 1997), web search (Mahoney and Chan, 2002), malware detection (Xu et al., 2016) and ad-blocking (Tramèr et al., 2018).

Recently, deep neural network (DNN) methods have shown their superiority in many fields such as image classification (LeCun et al., 2015), object detection (Redmon et al., 2016) and speech synthesis (Van den Oord et al., 2016). These methods are widely deployed as products or services to make human life more convenient. However, studies have highlighted the vulnerability of DNNs to adversarial perturbations (Goodfellow et al., 2015a; Szegedy et al., 2013). In other words, well-trained models can be easily fooled when adding imperceptible noise to the input data. The risk can be amplified to become a severe security problem if the model is deployed

for safety-critical or security-sensitive applications such as autonomous driving, face recognition, malware detection and security screening.

An effective way to robustify machine learning models is to exploit the information of these malicious inputs. Models are not only trained on normal data but also trained on adversarial examples (Dalvi et al., 2004; Huang et al., 2011; Madry et al., 2018). Therefore, an efficient way to generate adversarial examples is necessary to train robust models.

In general, adversarial attacks can be formulated as an optimization problem. The objective function of the problem is designed so that the adversarial perturbation is minimized while the attack goal is achieved. The majority of the previous work on attacking DNNs assumes a "white-box" setting, in which the complete information of the model, such as model architecture and model parameters, is known. This essentially allows an attacker to evaluate the target network from both the input-to-output direction and the output-to-input direction. In particular, an attacker can "query" the network by applying a test image to the network and observe the corresponding response produced at the outputs. An attacker is also allowed to perform "back-propagation" on the network; the output gradients can propagate through the network to the input image. A network under the white-box setting is considered relatively easy to attack since all the information is transparent and the optimization problem can be solved efficiently through optimization algorithms such as gradient descent or its variants. Existing powerful white-box methods include the fast gradient sign method (Kurakin et al., 2017), Carlini and Wagner's (C&W) attack (Carlini and Wagner, 2017) and the elastic-net attack (Chen et al., 2018).

Knowing the entire model information is a strong assumption. In practice, attackers have limited knowledge of the model. The only information available to the attackers are the inputs and the corresponding outputs of the model. Under this black-box setting, the gradient-based methods are impractical. In Papernot et al.

(2017), the authors proposed to train a substitute model through model queries, perform white-box adversarial attacks on the substitute model and utilize the transferability of the adversarial examples (Papernot et al., 2016) to attack the target model. However, training a substitute model for DNN is a challenging problem owing to its nature of high dimensionality and non-linearity. The performance of the black-box attacks highly depends on the transferability of the substitute model to the target model and is often unsatisfactory. Authors in Chen et al. (2017); Nitin Bhagoji et al. (2018) proposed to estimate the coordinate-wise gradients directly through model queries. The coordinate-wise gradient estimation considerably increases the attack success rate but requires an excessive number of queries to the target model. Even though several techniques such as importance sampling and random feature grouping are applied to accelerate the optimization process, the number of queries is still unreasonable.

In this work, we propose the Autoencoder-based Zeroth Order Optimization Method, AutoZOOM, to improve the query efficiency of gradient-estimation and gradient-descent-based black-box attacks. One unique feature of AutoZOOM is the use of reduced attack dimensions when mounting black-box attacks, which is an unlabeled data-driven technique (autoencoder) for attack acceleration and has not been studied thoroughly in existing attacks. AutoZOOM is a general method so that it can be applied to a wide variety of models. In this work, we demonstrate the abilities of AutoZOOM on two commonly seen machine learning tasks: classification and regression. We use classification examples to illustrate the efficiency of AutoZOOM. The regression example is used to show the general usage of AutoZOOM as well as the fact that adversarial examples are indeed an issue in the field of machine learning and that we should take adversarial inputs into account so as to enhance the prediction performance.

5.1 AutoZOOM

Considering the high-dimensional covariate $x \in \mathbb{R}^D$ and the corresponding low-dimensional response $y \in \mathbb{R}^L$, we aim to generate adversarial images that are visually similar to x but that would result in model failure. Model failure indicates that the model would produce the wrong predicted class under a classification scenario or that it would result in large prediction errors (e.g., PMSE). A set of allowed perturbations $\mathcal{S} \subseteq \mathbb{R}^D$ is introduced to data point x so as to control the difference between the adversarial image and the original input data. Given a loss function $\mathcal{L}(x, y)$, the generation of adversarial examples can be formulated as follows:

$$\max_{\epsilon \in \mathcal{S}} \mathcal{L}(x + \epsilon, y) \tag{5.1}$$

where $\mathcal{S}$ is the "threat model" defined as $S = \{\epsilon : \|\epsilon\|_p / D \leq \delta\}$ $(p \geq 1)$. In other words, we would like to find a small perturbation under ℓ-p norm that would lead to a large loss.

Depending on the task of the target model, there are two kinds of problem formulations. One is for classification purposes and the other is for regression purposes.

Regression

Under a regression framework such as GLLiM, the adversarial examples are the ones that will result in large prediction loss. Here we use prediction mean squared error to evaluate the prediction performance. Denote the regression function $F_R : \mathbb{R}^D \mapsto \mathbb{R}^L$. Let (x, y) denote the natural input x and its ground-truth response y. The prediction loss can be defined as:

$$\mathcal{L}(x, y) = ||F_R(x) - y||_2^2 \tag{5.2}$$

With a Lagrangian multiplier, solving Equation (5.1) is equivalent to:

$$\max_{\epsilon} \left\{ ||F_R(x+\epsilon) - y||_2^2 - \lambda ||\epsilon||_2^2 \right\}, \tag{5.3}$$

where λ is the penalty term that would satisfy the constraint of $\mathcal{S}$. In practice, a suitable value of λ could be found by binary search.

Classification

The task of classification also falls into the high-to-low mapping setting. For a high-dimensional sample $x \in \mathbb{R}^D$, we aim to predict its scores belonging to L classes. Two types of classification attacks are discussed. One is the so-called targeted attack, which indicates that there is a specific class label an attacker desires to change to. As for an untargeted attack, as long as the predicted class is different from the original one, the attack is considered as success. Considering the classification function $F_C : \mathbb{R}^D \mapsto \mathbb{R}^L$ that, similarly, takes D-dimensional data as an input and yields a vector for prediction scores of L classes, we intend to generate adversarial examples from a data pair (x, y), where x is the input data and y is the ground-truth of the class label under one-hot encoding. For a targeted attack, we aim to alter x such that it is classified as class $y' \neq y$. We next define the loss function for classification as follows:

$$\mathcal{L}(x, y') = \max \left[\max_{j, j \neq y'} (F_c(x))_j - (F_c(x))_{y'}, 0 \right]. \tag{5.4}$$

Here we use subscripts $(\cdot)_j$ to denote the j-th element of the predicted scores. If the j-th element is the largest, the model will classify the input as class j. The loss function is designed to be zero if the attack succeeds. Namely, the loss function becomes zero if the score of being predicted as the targeted class y' is larger than that of other classes.

The generation of the adversarial example can be formulated as :

$$\min_{\epsilon} \left\{ \max \left[\max_{j, j \neq y'} (F_c(x+\epsilon))_j - (F_c(x+\epsilon))_{y'}, 0 \right] + \lambda ||\epsilon||_2^2 \right\}. \tag{5.5}$$

5.2 Efficient mechanism for gradient estimation

There are two major contributions of this work. First, we adopt the dimension reduction technique to accelerate the attack process. We attack in the reduced space and scale up the perturbation to the original size. This technique allows us to reduce the query counts effectively. The other technique is to estimate the gradient vector. In contrast to coordinate-wise gradient estimation, the gradient vector is much more efficient since the gradient of the whole image is updated. As a comparison, to estimate the gradient of the whole image using the coordinate-wise method, it requires $W \times H \times C$ numbers of calculation, where W is the width of the image, H is the height of the image and C is the number of the color channels.

5.2.1 Random vector based gradient estimation

As a first attempt to enable gradient-free black-box attacks on DNNs, the authors in Chen et al. (2017) use the symmetric difference quotient method (Lax and Terrell, 2014) to evaluate the gradient $\frac{\partial f(x)}{\partial x_i}$ of the i-th component by

$$g_i = \frac{f(x + he_i) - f(x - he_i)}{2h} \approx \frac{\partial f(x)}{\partial x_i} \tag{5.6}$$

using a small h. Here e_i denotes the i-th elementary basis. Albeit contributing to powerful black-box attacks and being applicable to large networks like ImageNet, the nature of coordinate-wise gradient estimation step in Equation (5.6) must incur an enormous amount of model queries and is hence not query-efficient. For example, the ImageNet dataset has $D = 299 \times 299 \times 3 \approx 270,000$ input dimensions,

rendering coordinate-wise zeroth order optimization based on gradient estimation query-inefficient.

To improve query efficiency, we dispense with coordinate-wise estimation and instead propose a scaled random full gradient estimator of $\nabla f(x)$, defined as

$$g = b \cdot \frac{f(x + \beta u) - f(x)}{\beta} \cdot u, \tag{5.7}$$

where $\beta > 0$ is a smoothing parameter, u is a unit-length vector that is uniformly drawn at random from a unit Euclidean sphere, and b is a tunable scaling parameter that balances the bias and variance trade-off of the gradient estimation error.

Averaged random gradient estimation.

To effectively control the error in gradient estimation, we consider a more general gradient estimator, in which the gradient estimate is averaged over q random directions $\{u_j\}_{j=1}^{q}$. That is,

$$\overline{g} = \frac{1}{q} \sum_{j=1}^{q} g_j, \tag{5.8}$$

where g_j is a gradient estimate defined in Equation (5.7) with $u = u_j$. The use of multiple random directions can reduce the variance of $\overline{g}$ in Equation (5.8) for convex loss functions (Duchi et al., 2015; Liu et al., 2018).

Below we establish an error analysis of the averaged random gradient estimator in Equation (5.8) for studying the influence of the parameters b and q on estimation error and query efficiency.

Theorem 5.1. *Assume $f : \mathbb{R}^d \mapsto \mathbb{R}$ is differentiable and its gradient $\nabla f(\cdot)$ is L-Lipschitz.*[1] *Then the mean squared estimation error of $\overline{g}$ in (5.8) is upper bounded*

[1] A function $W(\cdot)$ is L-Lipschitz if $\|W(w_1) - W(w_2)\|_2 \leq L\|w_1 - w_2\|_2$ for any w_1, w_2. For DNNs with ReLU activations, L can be derived from the model weights as in Szegedy et al. (2013).

by

$$\mathbb{E}\|\overline{g} - \nabla f(\mathbf{x})\|_2^2 \leq 4(\frac{b^2}{D^2} + \frac{b^2}{Dq} + \frac{(b-D)^2}{D^2})\|\nabla f(\mathbf{x})\|_2^2 + \frac{2q+1}{q} b^2 \beta^2 L^2. \tag{5.9}$$

Proof. The proof is given in Appendix B. □

Here we highlight important implications based on Theorem 5.1: (i) The error analysis holds when f is *non-convex*; (ii) In DNNs, the true gradient ∇f can be viewed as the numerical gradient obtained via back-propagation; (iii) For any fixed b, selecting a small β (e.g., we set $\beta = 1/D$ in AutoZOOM) can effectively reduce the last error term in Equation (5.9), and we therefore focus on optimizing the first error term; (iv) The first error term in Equation (5.9) exhibits the influence of b and q on the estimation error, and is independent of β. We further elaborate on (iv) as follows. Fixing q and letting $\eta(b) = \frac{b^2}{D^2} + \frac{b^2}{Dq} + \frac{(b-D)^2}{D^2}$ be the coefficient of the first error term in Equation (5.9), then the optimal b that minimizes $\eta(b)$ is $b^* = \frac{Dq}{2q+D}$. For query efficiency, one would like to keep q small, which then implies $b^* \approx q$ and $\eta(b^*) \approx 1$ when the dimension D is large. On the other hand, when $q \to \infty$, $b^* \approx D/2$ and $\eta(b^*) \approx 1/2$, which yields a smaller error upper bound but is query-inefficient. We also note that by setting $b = q$, the coefficient $\eta(b) = \frac{b^2}{D^2} + \frac{b^2}{Dq} + \frac{(b-D)^2}{D^2} \approx 1$ and thus is independent of the dimension D and the parameter q.

Adaptive random gradient estimation.

Based on Theorem 5.1 and our error analysis, in AutoZOOM we set $b = q$ in Equation (5.7) and propose to use an adaptive strategy for selecting q. AutoZOOM uses $q = 1$ (i.e., the fewest possible model evaluations) to first obtain rough gradient estimates for solving Equation (5.1) until a successful adversarial image is found. After the initial attack success, it switches to using more accurate gradient estimates with $q > 1$ to

fine-tune the image quality. The trade-off between q (which is proportional to query counts) and distortion reduction will be investigated in Section 5.3.

5.2.2 Attack dimension reduction

Dimension-dependent convergence rate using gradient estimation. Different from the first-order convergence results, the convergence rate of zeroth order gradient descent methods has an additional multiplicative dimension-dependent factor D. In the convex loss setting the rate is $O(\sqrt{D/I})$, where I is the number of iterations (Nesterov and Spokoiny, 2017; Liu et al., 2018; Gao et al., 2014; Wang et al., 2018). The same convergence rate has also been found in the nonconvex setting (Ghadimi and Lan, 2013). The dimension-dependent convergence factor D suggests that vanilla black-box attacks using gradient estimations can be query inefficient when the (vectorized) image dimension D is large, due to the curse of dimensionality in convergence. This also motivates us to propose using an autoencoder to reduce the attack dimension and improve query efficiency in black-box attacks.

In AutoZOOM, we propose to perform random gradient estimation from a reduced dimension $D' < D$ to improve query efficiency. Specifically, as illustrated in Figure 5.1, the additive perturbation to an image x_0 is actually implemented through a "decoder" $Dec : \mathbb{R}^{D'} \mapsto \mathbb{R}^D$ such that $x = x_0 + Dec(\delta')$, where $\delta' \in \mathbb{R}^{D'}$. In other words, the adversarial perturbation $\delta \in \mathbb{R}^D$ to x_0 is in fact generated from a dimension-reduced space, with an aim of improving query efficiency due to the reduced dimension-dependent factor in the convergence analysis. AutoZOOM provides two modes for such a decoder Dec:

- An autoencoder (AE) trained on unlabeled data that are different from the training data to learn reconstruction from a dimension-reduced representation. The encoder $Enc(\cdot)$ in an AE compresses the data to a low-dimensional latent space

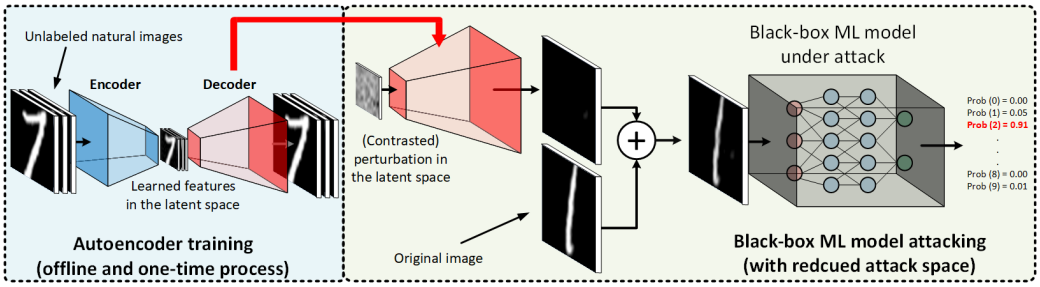


Figure 5.1: Illustration of attack dimension reduction through a "decoder" in AutoZOOM for improving query efficiency in black-box attacks. The decoder has two modes: (i) an autoencoder (AE) trained on unlabeled natural images that are different from the attacked images and training data; (ii) a simple bilinear image resizer (BiLIN) that is applied channel-wise to extrapolate low-dimensional features to the original image dimension (width $\times$ height). In the latter mode, no additional training is required.

and the decoder $Dec(\cdot)$ reconstructs an example from its latent representation. The weights of an AE are learned to minimize the average L_2 reconstruction error. Note that training such an AE for black-box adversarial attacks is one-time and is entirely offline (i.e., no model queries needed). Our proposal of AE is motivated by the insightful findings in Goodfellow et al. (2015b) that a successful adversarial perturbation is highly relevant to some human-imperceptible noise patterns resembling the shape of the target class, known as "shadow." Since a decoder in AE learns to reconstruct data from latent representations, it can also provide distributional guidance for mapping adversarial perturbations to generate these shadows.

- A simple channel-wise bilinear image resizer (BiLIN) that scales a small image to a large image via bilinear extrapolation. Note that no additional training is required for BiLIN.

Algorithm 5.1 AutoZOOM for black-box attacks on DNNs

Input : Black-box DNN model F, original example x_0, attack objective $\mathcal{L}(\cdot,\cdot)$, decoder $Dec(\cdot) \in \{\text{AE}, \text{BiLIN}\}$, initial coefficient λ_{ini}, query budget Q

Output : Least distorted successful adversarial example

while *query count* $\leq Q$ **do**

1. Exploration: use $x = x_0 + Dec(\delta')$ and apply the random gradient estimator in Equation (5.8) with $q = 1$ to the downstream optimizer (e.g., ADAM) for solving Equation (5.1) until an initial attack is found.

2. Exploitation (post-success stage): continue to fine-tune the adversarial perturbation $Dec(\delta')$ for solving Equation (5.1) while setting $q \geq 1$ in Equation (5.8).

end

We also note that for any reduced dimension D', the setting $b^* = q$ is optimal in terms of minimizing the corresponding estimation error from Theorem 5.1, despite the fact that the gradient estimation errors of different reduced dimensions cannot be directly compared. In Section 5.3 we will report the superior query efficiency in black-box attacks achieved with the use of AE or BiLIN as the decoder, and discuss the benefit of attack dimension reduction.

5.2.3 AutoZOOM algorithm

Algorithm 5.1 summarizes the AutoZOOM framework to query-efficient black-box attacks on DNNs. We also note that AutoZOOM is a general acceleration tool that is compatible with any gradient-estimation-based black-box adversarial attack obeying the attack formulation in Equation (5.1). It also has some theoretical estimation error guarantees and query-efficient parameter selection based on Theorem 5.1. The details on adjusting the regularization coefficient λ and the query parameter q based

on run-time model evaluation results will be discussed in Section 5.3.

5.3 Numerical results

Under the classification scenario, the definition of a successful attack is straightforward. We can check the classification results of the generated outcomes. If a generated sample can fool the classifier (i.e. the classification result changes), the attack is a success. We use the classification scenario to illustrate the efficiency of AutoZOOM in terms of query counts. Results for AutoZOOM attacking the regression application will be demonstrated in Chapter VI.

5.3.1 Experimental setup

We compare **AutoZOOM-AE** (D = AE) and **AutoZOOM-BiLIN** (D = BiLIN) with two different baselines: (i) standard **ZOO** implementation[2] with bilinear scaling (same as BiLIN) for dimension reduction; (ii) **ZOO+AE**, which is ZOO with AE. Note that all attacks indeed generate adversarial perturbations based on the same reduced attack dimensions.

We assess the performance of different attack methods on several representative benchmark datasets, including MNIST (LeCun et al., 1998), CIFAR-10 (Krizhevsky, 2009) and ImageNet (Russakovsky et al., 2015). MNIST contains digit numbers from zero to nine. It is a dataset with a relatively simple image structure. The image is under grayscale, and thus there is only one color channel. The CIFAR-10 dataset contains small-size colorful images. There are 10 classes with a somewhat complexed image structure. The width and the height of the image are both 32 pixels and there are three color channels. ImageNet is a collection of real-life photos. There are 1000 labels with more complicated image structures. Thus it would be challenging to modify an image to fit a certain class. We use MNIST and CIFAR-10 to illustrate the

[2] https://github.com/huanzhang12/ZOO-Attack

capability and effectiveness of AutoZOOM. Imagenet is used to show the superiority of AutoZOOM for sophisticated real-world examples. For MNIST and CIFAR-10, we use the same DNN image classification models[3] as in (Carlini and Wagner, 2017). For ImageNet, we use the Inception-v3 model (Szegedy et al., 2016). All experiments were conducted using TensorFlow Machine-Learning Library (Abadi et al., 2016) on machines equipped with an Intel Xeon E5-2690v3 CPU and an Nvidia Tesla K80 GPU.

All attacks used ADAM (Kingma and Ba, 2015) for solving Equation (5.1) with their estimated gradients and the same initial learning rate 2×10^{-3}. On MNIST and CIFAR-10, all methods adopt 1,000 ADAM iterations. On ImageNet, ZOO and ZOO+AE adopt 20,000 iterations, whereas AutoZOOM-BiLIN and AutoZOOM-AE adopt 100,000 iterations. Note that due to different gradient estimation methods, the query counts (i.e., the number of model evaluations) per iteration of a black-box attack may vary. ZOO and ZOO+AE use the parallel gradient update of Equation (5.6) with a batch of 128 pixels, yielding 256 query counts per iteration. AutoZOOM-BiLIN and AutoZOOM-AE use the averaged random full gradient estimator in Equation (5.8), resulting in $q+1$ query counts per iteration. For a fair comparison, the query counts are used for performance assessment.

The performances of different methods are evaluated using the metrics described below:

- **Query reduction ratio:** We use the mean query counts of ZOO with the smallest λ_{ini} as the baseline for computing the query reduction ratio of other methods and configurations.

- **TPR and initial success:** We report the true positive rate (TPR), which measures the percentage of successful attacks fulfilling a pre-defined constraint ℓ on the normalized (per-pixel) L_2 distortion, as well as the query counts of their

[3] https://github.com/carlini/nn_robust_attacks

first successes. We also report the per-pixel L_2 distortions of initial successes, where an initial success refers to the first query count that finds a successful adversarial example.

Post-success fine-tuning. When implementing AutoZOOM in Algorithm 5.1, on MNIST and CIFAR-10 we find that AutoZOOM without fine-tuning (i.e., $q = 1$) already yields similar distortion as ZOO. We note that ZOO can be viewed as coordinate-wise fine-tuning and is thus query-inefficient. On ImageNet, we will investigate the effect of post-success fine-tuning on reducing distortion.

Autoencoder Training. In AutoZOOM-AE, we use convolutional autoencoders for attack dimension reduction, which are trained on unlabeled datasets that are different from the training dataset and the attacked natural examples.

The implementation details are given in Table B.1. Note that the autoencoder designed for ImageNet uses bilinear scaling to transform the data size from $299 \times 299 \times Dep$ to $128 \times 128 \times Dep$, and also back from $128 \times 128 \times Dep$ to $299 \times 299 \times Dep$. This is to allow easy processing and handling for the autoencoder's internal convolutional layers. The normalized mean squared errors of our autoencoder trained on MNIST, CIFAR-10 and Imagenet are 0.0027, 0.0049 and 0.0151, respectively, which lies within a reasonable range of compression loss.

On MNIST, the convolutional autoencoder (CAE) is trained on 50,000 randomly selected hand-written digits from the MNIST8M dataset.[4] On CIFAR-10, the CAE is trained on 9,900 images selected from its test dataset. The remaining images are used in black-box attacks. On ImageNet, all the attacked natural images are from 10 randomly selected image labels, and these labels are also used as the candidate attack targets. The CAE is trained on about 9000 images from these classes. The architectures for all of the autoencoders used in AutoZOOM are shown in Table B.1.

To adjust the regularization coefficient λ in Equation (5.1), in all methods we set

[4] http://leon.bottou.org/projects/infimnist

its initial value $\lambda_{\text{ini}} \in \{0.1, 1, 10\}$ on MNIST and CIFAR-10, and set $\lambda_{\text{ini}} = 10$ on ImageNet. Furthermore, for balancing the distortion Dist and the attack objective Loss in Equation (5.1), we use a *dynamic switching* strategy to update λ during the optimization process. Per every S iteration, λ is multiplied by 10 times the current value if the attack has never been successful. Otherwise, it divides its current value by 2. On MNIST and CIFAR-10, we set $S = 100$. On ImageNet, we set $S = 1,000$. At the instance of initial success, we also reset $\lambda = \lambda_{\text{ini}}$ and the ADAM parameters to the default values, as doing so can empirically reduce the distortion for all attack methods.

5.3.2 Black-box attacks on MNIST

For MNIST, we randomly select 50 correctly classified images from their test sets, and perform targeted attacks on these images. Since there are 10 classes in MNIST, each selected image is attacked 9 times, targeting all but its true class. For all attacks, the ratio of the reduced attack-space dimension to the original one (i.e., D'/D) is 25%.

Table 5.1 shows the performance evaluation on MNIST with various values of λ_{ini}, the initial value of the regularization coefficient λ in Equation (5.1). We use the performance of ZOO with $\lambda_{\text{ini}} = 0.1$ as a baseline for comparison. For example, with $\lambda_{\text{ini}} = 0.1$ and 10, the mean query counts required by AutoZOOM-AE to attain an initial success are reduced by **93.21%** and **98.57%**, respectively. One can also observe that allowing larger λ_{ini} generally leads to fewer mean query counts at the price of slightly increased distortion for the initial attack. The noticeable huge difference in the required attack query counts between AutoZOOM and ZOO/ZOO+AE validates the effectiveness of our proposed random full gradient estimator in Equation (5.7), which dispenses with the coordinate-wise gradient estimation in ZOO but still retains comparable true positive rates, thereby greatly improving query efficiency.

Method	λ_{ini}	Attack success rate (ASR)	Mean query count (initial success)	Mean query count reduction ratio (initial success)	Mean per-pixel L_2 distortion (initial success)	True positive rate (TPR)	Mean query count with per-pixel L_2 distortion ≤ 0.004
ZOO	0.1	99.44%	35,737.60	0.00%	3.50×10^{-3}	96.76%	47,342.85
	1	99.44%	16,533.30	53.74%	3.74×10^{-3}	97.09%	31,322.44
	10	99.44%	13,324.60	62.72%	4.85×10^{-3}	96.31%	41,302.12
ZOO+AE	0.1	99.67%	34,093.95	4.60%	3.43×10^{-3}	97.66%	44,079.92
	1	99.78%	15,065.52	57.84%	3.72×10^{-3}	98.00%	29,213.95
	10	99.67%	12,102.20	66.14%	4.66×10^{-3}	97.66%	38,795.98
AutoZOOM-BiLIN	0.1	99.89%	2,465.95	93.10%	4.51×10^{-3}	96.55%	3,941.88
	1	99.89%	879.98	97.54%	4.12×10^{-3}	97.89%	2,320.01
	10	99.89%	612.34	98.29%	4.67×10^{-3}	97.11%	4,729.12
AutoZOOM-AE	0.1	**100.00%**	2,428.24	**93.21%**	4.54×10^{-3}	96.67%	3,861.30
	1	**100.00%**	729.65	**97.96%**	4.13×10^{-3}	96.89%	1,971.26
	10	**100.00%**	510.38	**98.57%**	4.67×10^{-3}	97.22%	4,855.01

Table 5.1: Performance evaluation of black-box targeted attacks on MNIST.

5.3.3 Black-box attacks on CIFAR-10

As for CIFAR-10, 50 correctly classified testing images are selected as well. We then perform targeted attacks on these selected images. Similar to MNIST, there are 10 classes in the CIFAR-10 dataset. Thus, each selected image is attacked 9 times, targeting other classes other than its true one. For all attacks, we reduce the original attack-space dimension from $32 \times 32 \times 3$ to $8 \times 8 \times 3$ ($D'/D = 6.25\%$).

For CIFAR-10, we report similar query efficiency improvements as displayed in Table 5.2. In particular, comparing the two query-efficient black-box attack methods (AutoZOOM-BiLIN and AutoZOOM-AE), we find that AutoZOOM-AE is more query-efficient than AutoZOOM-BiLIN, but at the cost of an additional AE training step. AutoZOOM-AE achieves the highest attack success rates (ASRs) and mean query reduction ratios for different values of λ_{ini}. In addition, their true positive rates (TPRs) are similar but AutoZOOM-AE usually takes fewer query counts to reach the same L_2 distortion. We note that when $\lambda_{\text{ini}} = 10$, AutoZOOM-AE has a higher TPR but also needs slightly more mean query counts than AutoZOOM-BiLIN to reach the same L_2 distortion. This suggests that there are some adversarial examples for which it is difficult for a bilinear resizer to reduce their post-success distortions but which can be handled by an AE.

Method	λ_{ini}	Attack success rate (ASR)	Mean query count (initial success)	Mean query count reduction ratio (initial success)	Mean per-pixel L_2 distortion (initial success)	True positive rate (TPR)	Mean query count with per-pixel L_2 distortion ≤ 0.0015
ZOO	0.1	97.00%	25,538.43	0.00%	5.42×10^{-4}	100.00%	25,568.33
	1	97.00%	11,662.80	54.33%	6.37×10^{-4}	100.00%	11,777.18
	10	97.00%	10,015.08	60.78%	8.03×10^{-4}	100.00%	10,784.54
ZOO+AE	0.1	99.33%	19,670.96	22.98%	4.96×10^{-4}	100.00%	20,219.42
	1	99.00%	5,793.25	77.32%	6.83×10^{-4}	99.89%	5,773.24
	10	99.00%	4,892.80	80.84%	8.74×10^{-4}	99.78%	5,378.30
AutoZOOM-BiLIN	0.1	99.67%	2,049.28	91.98%	1.01×10^{-3}	98.77%	2,112.52
	1	99.67%	813.01	96.82%	8.25×10^{-4}	99.22%	1,005.92
	10	99.33%	623.96	97.56%	9.09×10^{-4}	98.99%	835.27
AutoZOOM-AE	0.1	**100.00%**	1,523.91	**94.03%**	1.20×10^{-3}	99.67%	1,752.45
	1	**100.00%**	332.43	**98.70%**	1.01×10^{-3}	99.56%	345.62
	10	**100.00%**	259.34	**98.98%**	1.15×10^{-3}	99.67%	990.61

Table 5.2: Performance evaluation of black-box targeted attacks on CIFAR-10.

5.3.4 Black-box attacks on ImageNet

We selected 50 correctly classified images from the ImageNet test set to perform random targeted attacks and set $\lambda_{\text{ini}} = 10$ and the attack dimension reduction ratio to 1.15%.

As described in Algorithm 5.1, adaptive random gradient estimation is integrated into AutoZOOM, offering a quick initial success in attack generation followed by a fine-tuning process to effectively reduce the distortion. This is achieved by adjusting the gradient estimate averaging parameter q in Equation (5.8) in the post-success stage. In general, averaging over more random directions (i.e., setting larger q) tends to better reduce the variance of gradient estimation error, but at the cost of increased model queries. Figure 5.2 (a) shows the mean distortion against query counts for various choices of q in the post-success stage. The results suggest that setting some small q but $q > 1$ can further decrease the distortion at the converged phase when compared with the case of $q = 1$. Moreover, the refinement effect on distortion empirically saturates at $q = 4$, implying a marginal gain beyond this value. These findings also demonstrate that our proposed AutoZOOM indeed strikes a balance between distortion and query efficiency in black-box attacks.

The results are summarized in Table 5.3. Note that compared to ZOO, AutoZOOM-AE can significantly reduce the query count required to achieve an initial success by

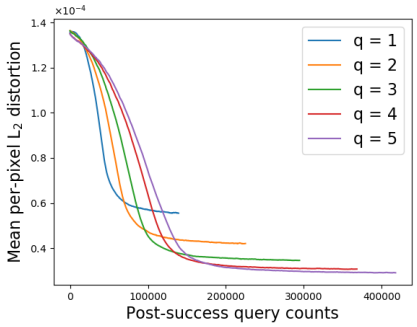
Method	Attack success rate (ASR)	Mean query count (initial success)	Mean query count reduction ratio (initial success)	Mean per-pixel L_2 distortion (initial success)	True positive rate (TPR)	Mean query count with per-pixel L_2 distortion ≤ 0.0002
ZOO	76.00%	2,226,405.04 (2.22M)	0.00%	4.25×10^{-5}	100.00%	2,296,293.73
ZOO+AE	92.00%	1,588,919.65 (1.58M)	28.63%	1.72×10^{-4}	100.00%	1,613,078.27
AutoZOOM-BiLIN	**100.00%**	14,228.88	99.36%	1.26×10^{-4}	100.00%	15,064.00
AutoZOOM-AE	**100.00%**	13,525.00	**99.39%**	1.36×10^{-4}	100.00%	14,914.92

Table 5.3: Performance evaluation of black-box targeted attacks on ImageNet

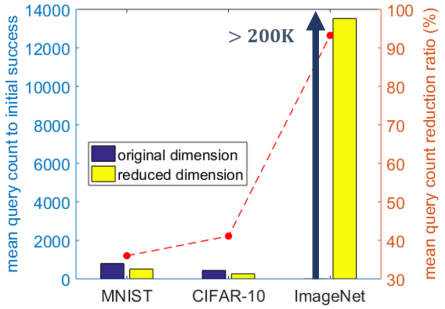
99.39% (or 99.35% to reach the same L_2 distortion), which is a remarkable improvement since this means a reduction of more than *2.2 million* model queries given the fact that the dimension of ImageNet ($\approx$ 270K) is much larger than that of MNIST and CIFAR-10.

5.3.5 Dimension reduction and query efficiency

In addition to the motivation from the $O(\sqrt{D/I})$ convergence rate in zeroth-order optimization (Section 5.2.2), as a sanity check, we corroborate the benefit of attack dimension reduction to query efficiency in black-box attacks by comparing AutoZOOM (here we use D = AE) with its alternative operating on the original (non-reduced) dimension (i.e., $\delta' = D(\delta') = \delta$). Tested on all three datasets and aforementioned settings, Figure 5.2 (b) shows the corresponding mean query count to initial success and the mean query reduction ratio when $\lambda_{\text{ini}} = 10$ in all three datasets. When compared to the attack results of the original dimension, attack dimension reduction through AutoZOOM reduces query counts on MNIST by roughly 35-40% and CIFAR-10 and by at least 95% on ImageNet. This result highlights the importance of dimension reduction for query-efficient black-box attacks. For example, without dimension reduction, the attack on the original ImageNet dimension cannot even be successful within the query budget ($Q = 200K$ queries).



(a) Post-success distortion refinement



(b) Dimension reduction v.s. query efficiency

Figure 5.2: (a) After initial success, AutoZOOM (here D = AE) can further decrease the distortion by setting $q > 1$ in (5.8) to trade more query counts for smaller distortion in the converged stage, which saturates at $q = 4$. (b) Attack dimension reduction is crucial to query-efficient black-box attacks. When compared to black-box attacks on the original dimension, dimension reduction through AutoZOOM-AE reduces query counts by roughly 35-40% on MNIST and CIFAR-10 and by at least 95% on ImageNet.

5.4 Conclusion

The attack gain in AutoZOOM-AE versus AutoZOOM-BiLIN could sometimes be marginal, while we also note that there is room for improving AutoZOOM-AE by exploring different AE models. However, we advocate AutoZOOM-BiLIN as a practically ideal candidate for query-efficient black-box attacks when testing model robustness, due to its easy-to-mount nature and its not having any additional training cost. While learning effective low-dimensional representations of legitimate images is still a challenging task, black-box attacks using significantly fewer degrees of freedom (i.e., reduced dimensions), as demonstrated in this paper, are certainly plausible, leading to new implications for model robustness.

CHAPTERS VI

Prediction when Input Signals are Corrupted or Adversarially Perturbed

Prediction efficacy often relies on the prediction model being insensitive to outlying training data points. A common practice is to choose the predictive model to be robust to outliers. How to build such a predictive model has been extensively studied in different fields. Nevertheless, this practice puts a restriction on users' choice of predictive methods. Alternatively, when information about normal data exists, one can use a selected robust platform to preprocess or to screen new data entries. Users could then apply their preferred predictive methods to the processed new data. In this work, we propose a preprocessing method to handle two commonly seen types of abnormal data. The abnormal data is referred to as "corrupted" if a small portion of the data is contaminated. This could happen if the data collection process is defective. Another kind of abnormal data, adversarial data are malicious inputs that are designed to fool the model, which could be seen when users try to hack the system.

The preprocessing includes two stages. At the first stage, we determine the type of the query data. If the query data is normal, the predictive model can be applied to the data directly. On the other hand, if the query data is abnormal (corrupted or adversarial), corrections would be made at the second stage before applying the predictive model. Each stage corresponds to a building block. The *aberrant data detector*

is responsible for determining the type of query data. The *aberrant data corrector* would conduct data correction if needed. Based on the detection results, the aberrant data corrector would correct the query data using different strategies. Prediction is then performed on the preprocessed data using the user-preferred method.

Throughout this work, we use image data as an example to demonstrate the capability of the proposed preprocessing system. However, the preprocessing framework is not limited to image data. Following the description of abnormal data, corrupted images are ones with a small area of distortion. Images can be corrupted for different reasons. One possible cause is that the light source could be blocked when taking the picture which causes a shadow. Another possibility is that the images have deteriorated because of physical damage such as scratches or stains. The deteriorated area could be small but could severely downgrade the prediction performance. Adversarial images are another kind of abnormal data. Studies have shown that machine learning methods are vulnerable to adversarial perturbations (Dalvi et al., 2004). The success of deceiving object detectors in the real world (Eykholt et al., 2018; Kurakin et al., 2017) brings the issue to a new level since the vulnerability may cause irreparable loss. In Chapter V we discuss an efficient method to generate adversarial examples. We adopt the proposed method, AutoZOOM, in this chapter to generate adversarial testing images to evaluate the reliability of the proposed system.

6.1 Robust preprocessing system

The robust preprocessing system consists of two building blocks: the *aberrant data detector* and the *aberrant data corrector*. The aberrant data detector classifies the query data into three categories: normal, corrupted and adversarial. Based on the classification results, the aberrant data corrector performs different correction methods on the query data. No further process is applied if a sample is identified as normal. If a sample is classified as corrupted, we utilize the inverse regression learned

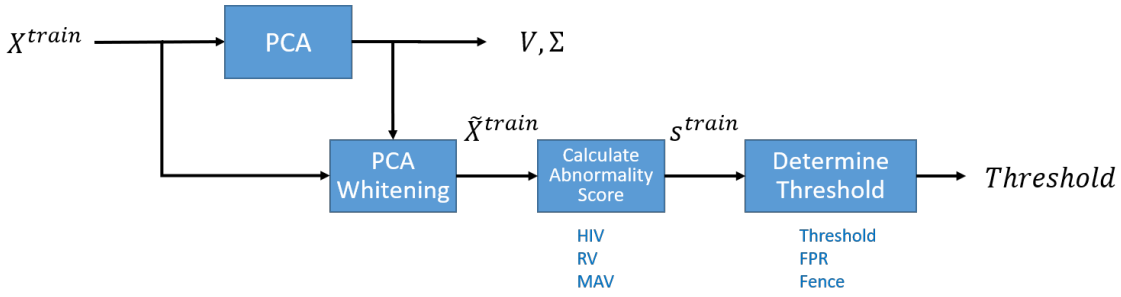


Figure 6.1: The flowchart of constructing the robust preprocessing system at the training stage.

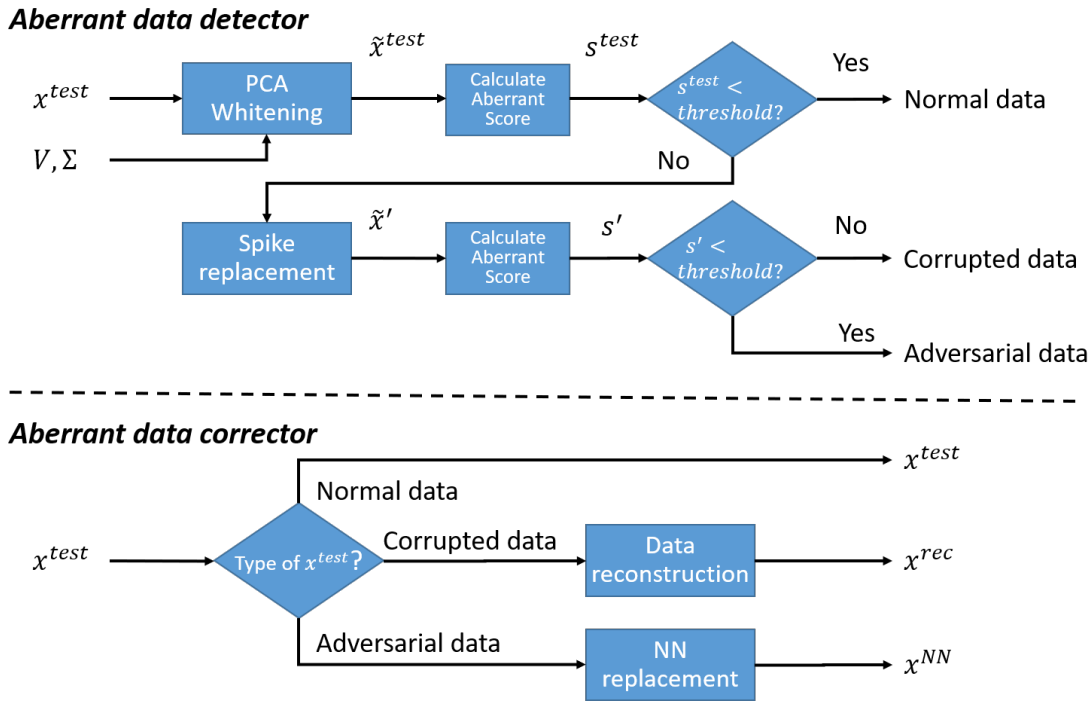


Figure 6.2: The flowchart of the robust preprocessing system at the testing stage.

under the GLLiM framework for data reconstruction. As for adversarial data, we use the nearest neighbors in the training dataset as surrogate for prediction.

6.1.1 Aberrant data detector

The aberrant data detector is responsible for classifying the query data into three categories. The detector is built on principal component analysis (PCA) whitening. To perform PCA whitening, we first calculate the singular vectors and singular values

of the training data. As a new testing data point comes in, we calculate its aberrant score and compare the score with a pre-defined *threshold*. If the aberrant score is greater than *threshold*, the testing data point is classified as abnormal (corrupted or adversarial). Otherwise, it is normal data. The mechanism can essentially differentiate normal data from abnormal data. We further extend the approach so that two kinds of abnormal data, corrupted data and adversarial data, can be identified separately.

The flowchart for building an aberrant data detector at the training stage is shown in Figure 6.1. We first center the training data around zero and perform the singular value decomposition. Denote $X^{train} \in \mathbb{R}^{N \times D}$ as the centered training dataset where N is the number of the training samples and D is the dimension of the sample. We decompose the training dataset as $X^{train} = U\Sigma V^{\top}$ where U is an $N \times N$ unitary matrix, Σ is an $N \times D$ diagonal matrix and V is a $D \times D$ unitary matrix. Letting $\tilde{x}_n \in \mathbb{R}^D$ be the normalized coefficients of the n-th sample in X^{train}, we have

$$\tilde{x}_{n,i} = \frac{x_n \cdot v_i}{\sigma_i}, \tag{6.1}$$

where v_i is the i-th column vector of matrix V and σ_i is the i-th singular value following in descending order.

PCA whitening projects a query image onto the principal components extracted from the training dataset. The projected coefficients are normalized by the singular values (σ_i) corresponding to the principal components, which we refer to as the normalized coefficients. These normalized coefficients can be used to detect abnormal images. As shown in Figure 6.3, the normalized coefficients for a normal image lie within a small range while those for an abnormal image might not. In particular, the scale of the high-indexed normalized coefficients for a problematic image could be large. For a normal image, even after the normalized coefficients are scaled by small

singular values, the resulting coefficients are still within a certain range. On the other hand, we would obtain larger coefficients from the last few principal components when they correspond to those being distorted or perturbed. These coefficients are those we observe in Figure 6.3 (b) and (c). Based on the normalized coefficients at high indexes, we can distinguish normal data from abnormal (corrupted and adversarial) data.

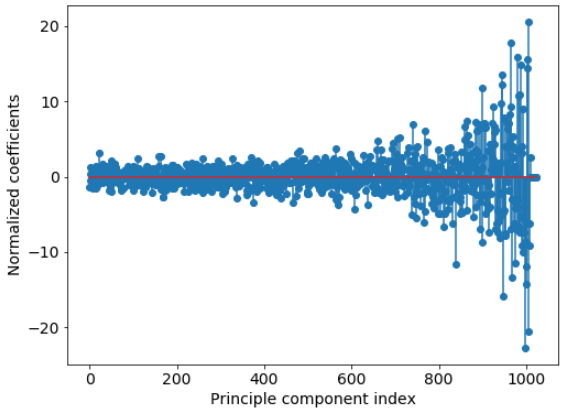
Calculate the aberrant score

In Hendrycks and Gimpel (2017), the authors calculated the variance of the high-indexed normalized coefficients as aberrant scores. Hereinafter, we refer to this method as high index variance (HIV). We denote P_{NC} as the portion of the high index normalized coefficients used to calculate the variance. As an example, if the data dimension (D) is 1024 and $P_{NC} = 10$, we would use the last 103 normalized coefficients (the 992-th to the 1024-th) to calculate the variances. We further consider other approaches to calculating aberrant scores. The rolling variance (RV) method would calculate the variances using the normalized coefficients in a sliding window. The length of the sliding window is defined by a parameter P_{window}, which is the portion of the consecutive normalized coefficients included in the sliding window. The aberrant score is the maximum value of all of the rolling variances. Finally, the maximum absolute value (MAV) method treats the normalized coefficients with the largest absolute values as the aberrant scores.

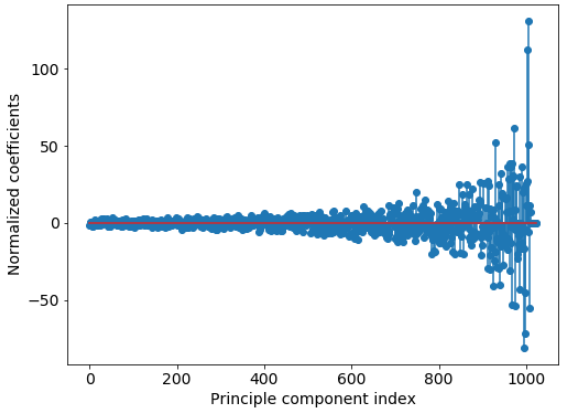
Determine the classification threshold

In Figure 6.1, we calculate the aberrant score s^{train} for each training sample. The next step is to determine the *threshold* for classification. Three methods are proposed. First, we can directly assign (DA) the *threshold*. The *threshold* is selected to reach the best detection results. Under this scenario, both normal and abnormal

(a)



(b)



(c)

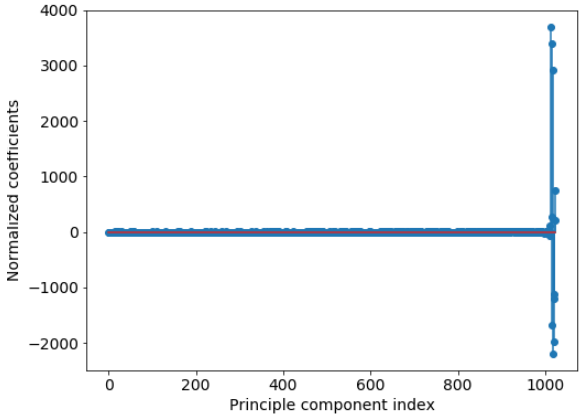


Figure 6.3: The normalized coefficients across different principal components of different image types in Figure 6.4: (a) normal image, (b) corrupted image and (c) adversarial image.

images are used. In the second approach, we propose to control the false positive rate, fpr, of wrongly detecting abnormal data. That is, given a false positive rate fpr, the $threshold$ is set to the $1 - fpr$ quantile of the aberrant scores obtained from the normal training data. Our last proposed approach is to locate the so-called "fence" which is commonly used in the boxplots. We calculate the mean, μ_{score} and the standard deviation, σ_{score} of the aberrant scores, and the fence is calculated as $\mu_{score} + M \times \sigma_{score}$, so that the distance between the fence and the mean is M times standard deviation. We note that the last two approaches do not utilize abnormal entries and can accommodate the scenarios when only normal data are available. These three methods are referred to as "DA", "FPR" and "Fence," respectively.

Differentiate corrupted and adversarial data

By comparing the aberrant score and the pre-determined $threshold$, we can classify the testing data as normal or abnormal. To further differentiate corrupted and adversarial samples, we take advantage of the following observations. In Figure 6.3, there are only few extreme normalized coefficients for adversarial data. On the other hand, no obvious spike appears on the normalized coefficients of corrupted data. Thus, we can remove the spikes and re-calculate the aberrant score again. If the classification result changes, we classify the query data as an adversarial entry, and otherwise as a corrupted entry.

The flowchart of the aberrant data detector at the testing stage is shown in Figure 6.2. For each testing data, $x^{test} \in \mathbb{R}^D$, we first calculate the aberrant score s^{test} using the results of SVD on the training dataset (V, Σ). The aberrant score is compared to $threshold$. If $s^{test} < threshold$, it is classified as a normal sample. Otherwise, it is an abnormal sample. We next find its nearest neighbor in the training dataset, denoted as x^{NN}, and calculate the normalized coefficients of x^{NN}, denoted as $\tilde{x}^{NN}$. We replace the normalized coefficients of $\tilde{x}^{test}$. Denoting $\tilde{x}'$ as the new normalized

coefficient vector after replacement, $\tilde{x}'$ is constructed as follows:

$$\tilde{x}'_i = \begin{cases} \tilde{x}_i^{NN} & \text{if } |\tilde{x}_i^{test}| > \textit{cutoff}, \\ \tilde{x}_i^{test} & \text{otherwise}, \end{cases} \tag{6.2}$$

where the subscript i denotes the i-th normalized coefficient of $\tilde{x}^{test}$, $\tilde{x}^{NN}$ and $\tilde{x}'$; *cutoff* is a pre-defined value to identify spike coefficients. The process described in Equation (6.2) is referred to as spike-replacement, where we replace the spike coefficients with normal ones. We then calculate the aberrant score of $\tilde{x}'$ as s' and compare s' to *threshold*. The testing sample is classified as an adversarial sample if the aberrant score is less than *threshold*, and as a corrupted sample otherwise.

6.1.2 Aberrant data corrector

Based on the classification results of the aberrant data detector, the aberrant data corrector would adopt different mechanisms. For the corrupted data, we would conduct data reconstruction, which utilizes the inverse regression of GLLiM. Recall that he inverse conditional density of GLLiM can be written as:

$$\sum_{k=1}^{K} p(X = x|Y = y, Z = k; \theta)p(Y = y|Z = k; \theta)p(Z = k; \theta)$$

By taking the expectation over the inverse conditional density, we can predict the covariate x for a given y as:

$$\mathbb{E}[X|Y = y] = \sum_{k=1}^{K} \frac{\pi_k N(y; c_k, \Gamma_k)}{\sum_{j=1}^{K} \pi_j N(y; c_j, \Gamma_j)}(A_k x + b_k). \tag{6.3}$$

First, a GLLiM model is built on the training dataset. Denote $GLLiM\text{-}Inv(\cdot)$ as the GLLiM inverse function in Equation (6.3). Given a response $y \in \mathbb{R}^L$, we can reconstruct the high-dimensional data x through $x = GLLiM\text{-}Inv(y)$. The

reconstruction of the corrupted data can be formulated as follows. Let

$$y^* = \arg\min_y Sim(x^{test}, GLLiM\text{-}Inv(y)), \tag{6.4}$$

where $Sim(\cdot, \cdot)$ is a function measuring the similarity between the testing data x^{test} and the reconstructed data, $GLLiM\text{-}Inv(y)$. Note that with x^{test} being a corrupted image, we leave out the corrupted pixels when measuring the similarity between the testing image and the reconstructed one. To achieve this goal, we design the similarity function using the truncated sum of squared differences. The similarity between two vectors a, b of dimension D for a given truncated quantile q is defined as follows:

$$Sim(a, b) = \sum_{i=1}^{D'} c'_i, \tag{6.5}$$

where we define c as a vector with its i-th entry $c_i = (a_i - b_i)^2$ and let c' be a permutation of c in ascending order, i.e. c'_i is the i-th smallest squared difference between elements in a and b. We define $D' = \lfloor D \times q \rfloor$ where $\lfloor \cdot \rfloor$ denotes the floor function. The truncated sum of squared differences is the summation over the first D' smallest squared differences. We will discuss details about the selection of q and the reconstruction results in Section 6.2.2. The reconstructed image is denoted as x^{rec} in Figure 6.2, which can be obtained by $x^{rec} = GLliM\text{-}Inv(y^*)$.

The output of the aberrant data corrector depends on the results of the detector. If the testing data is identified as normal, x^{test} would be provided directly. The output would be x^{rec} for a corrupted sample. As for an adversarial example, we use the nearest neighbor in the training dataset, x^{NN}, as the prediction surrogate since adversarial examples are close to the normal data manifold. After obtaining the preprocessed data from the corrector, we can apply the selected predictive method for prediction.

6.2 Numerical results

We use the face dataset for demonstration. The original images in the dataset are treated as normal data. We first discuss the generation of the corrupted and adversarial images and then demonstrate the performance of data reconstruction. We next discuss the detection performances under different combinations of settings. The prediction performances using three predictive models are presented to illustrate the efficacies of the proposed method for reducing the prediction errors.

6.2.1 Generation of abnormal images

The abnormal images are used for testing the robustness of the correction methods. We first separate the face dataset into a training dataset and a testing dataset. The training dataset contains 598 images and the rest of the 100 images are treated as testing samples. To generate corrupted images from the normal images, we randomly select an area. The maximum size of the area is set to be 4×16 ($pixel^2$). Compared to the original image size, 32×32, at most 1/16 of the image will be corrupted. Next, we set the pixel within the selected area to be masked (replaced with pixel values of black color), mimicking the occurrence of damage. Figure 6.4(b) shows an example of the corrupted testing images. Adversarial images are generated using AutoZOOM. Using the given GLLiM forward model discussed in Section 6.2.4, we generate adversarial testing data with the restriction of the normalized perturbation must be less than 0.001 (i.e. $||\epsilon||_2/D < 0.001$, see Chapter V for more details). The adversarial example is shown in Figure 6.4(c). In Figure 6.4(d), we plot the difference between the normal image and its adversarial counterpart. We can observe that the difference between the normal and the adversarial image is negligible.

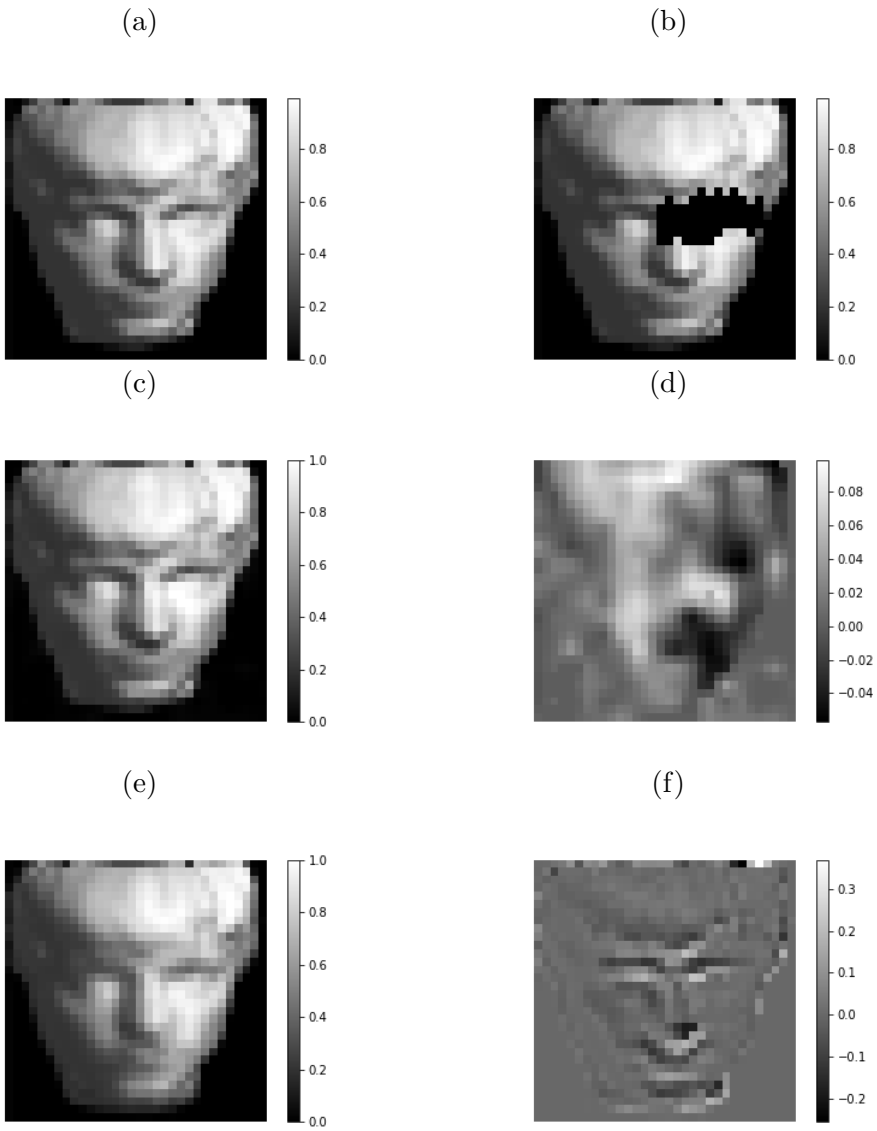


Figure 6.4: Examples of different kinds of images. (a) The normal image. (b) The corrupted image. (c) The adversarial image. (d) The difference between the adversarial image and the normal image. (e) The reconstructed image of (b). (f) The difference between the reconstructed image and the normal image.

6.2.2 Corrupted image reconstruction

In this section we demonstrate the ability of the preprocessing method to reconstruct the corrupted images. Note that we use a similarity function, $Sim(\cdot, \cdot)$, to evaluate the reconstruction of the data. To appropriately reconstruct the image, we use a truncated sum of squared differences between the reconstructed data and the testing data. Note that only a small number of the pixels are damaged in a cor-

rupted image. We only have to consider the reconstruction performance based on the normal pixels. To investigate how many pixels should be included when calculating the distance, we calculate the average squared differences of pixels between the corrupted image and their nearest neighbors in the training dataset. The differences are sorted and shown in Figure 6.5. We observe there is an abrupt change around the 90% quantile. Based on this analysis, we set $q = 0.9$. That is, we only calculate the similarity using 90% of the squared differences. The GLLiM inverse function is obtained with 20 clusters ($K = 20$) and 9 latent factors ($L_w = 9$). Figure 6.4 (e) shows the reconstructed image; the difference between the normal image (Figure 6.4 (a)) and the corrupted image (Figure 6.4(e)) is shown in Figure 6.4(f). Comparing the reconstructed image to the normal image, we can barely see the difference.

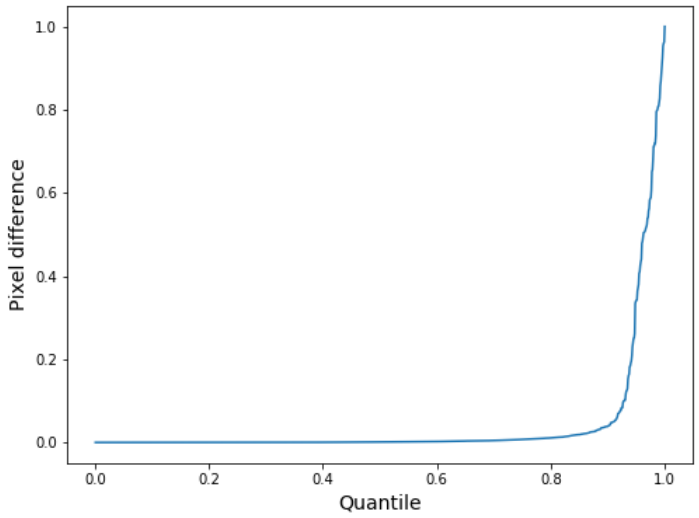


Figure 6.5: The sorted squared differences between corrupted images and their nearest neighbors in the training dataset.

6.2.3 Detection performance

The proposed data detector is built on a basic detector that can only differentiate normal and abnormal data. For the basic detector, users need to specify the portion

of the normalized coefficients used in the calculation of the aberrant score. We first perform a tuning parameter selection study on P_{NC}, P_{window} under different settings. Next, the detection performance of the data detector is presented.

When calculating the aberrant scores using a given method (HIV, RV or MAV), one needs to specify the portion of the normalized coefficients used in the calculation (P_{NV}) or the window size to calculate the rolling variance (P_{window}). We conduct a 10-fold cross-validation (CV) study on different combinations of aberrant score calculation and classification threshold determination to select the tuning parameters. The normal images in the CV training dataset are used to calculate the principal components and the singular values. Using these principal components and singular values, we calculate the normalized coefficients and aberrant scores of each CV testing sample, and compare the aberrant scores to *threshold*, which is determined by the specified classification threshold. The quality of the detection performances is shown by a CV study, in which we consider different combinations of aberrant-score calculation and classification threshold determination. The procedures are described below.

Calculate the aberrant score:

1. High index variance (HIV): We calculate the variance of the high index normalized coefficients as the aberrant score. The parameter P_{NC} specifies the portion of the high index normalized coefficients used for calculation. We study the detection performance when $P_{NC} = 5, 10, 20, 30, 40$.

2. Rolling variance (RV): The rolling variance is calculated on the normalized coefficients with the window size defined by the parameter P_{window}, and the aberrant score is the maximum value of the rolling variance. We investigate the detection performance when $P_{window} = 5, 10, 20, 30, 40$.

3. Maximum absolute value (MAV): The aberrant score is the largest absolute

value of the normalized coefficients. No extra tuning parameter is needed when using this method to calculate the aberrant score.

Determine the classification threshold

1. Directly assign (DA): When this method is adopted, we specify the threshold directly by setting it to be $10, 20, 30$.

2. False positive rate (FPR): By specifying the false positive rate, fpr, we will set the threshold as the $1-fpr$ quantile of the training aberrant scores and evaluate the detection performance on the *threshold*. We set $fpr = 0.05, 0.01, 0.005$ for performance evaluation.

3. Fence: We calculate the mean, μ_{score}, and the standard deviation, σ_{score}, of the aberrant scores. The fence is defined by a tuning parameter M, which is a multiplier of the standard deviation, and is calculated as $fence = \mu_{score} + M \times \sigma_{score}$. We use the fence as the threshold and conduct studies when $M = 2, 3, 4$.

When using the DA approach, we use the normal training data and the abnormal training data together to evaluate the classification accuracy. In each fold of CV, the aberrant data detector is first built based on normal training images. Next, we calculate the classification accuracy using the CV normal testing images and their abnormal counterparts (corrupted and adversarial). To make the number of normal data and abnormal data the same, we double-weighted the normal data. As an example, if there are 60 CV testing data, we use 60 CV normal testing data with weight fraction 50%, 60 CV corrupted testing data with weight fraction 25% and 60 CV adversarial data with weight fraction 25% to calculate the classification accuracy. Figure 6.6 shows the results for the DA approach. When more normalized coefficients are included to calculate the variance, i.e. large P_{NC} or large P_{window}, we obtain smaller variance. Thus, we can obtain better classification accuracy when the threshold is

small and P_{NC} (P_{window}) is large. We can obtain similar detection accuracy when $threshold = 10$ and $threshold = 20$ if P_{NC} and P_{window} are set to certain appropriate values. When $threshold = 30$, the detection accuracy is slightly lower if we use high-indexed normalized coefficients to calculate the variance. Comparing the results of using RV to those obtained from HIV, we can obtain better detection results using RV. The selected parameters are shown in Table 6.1.

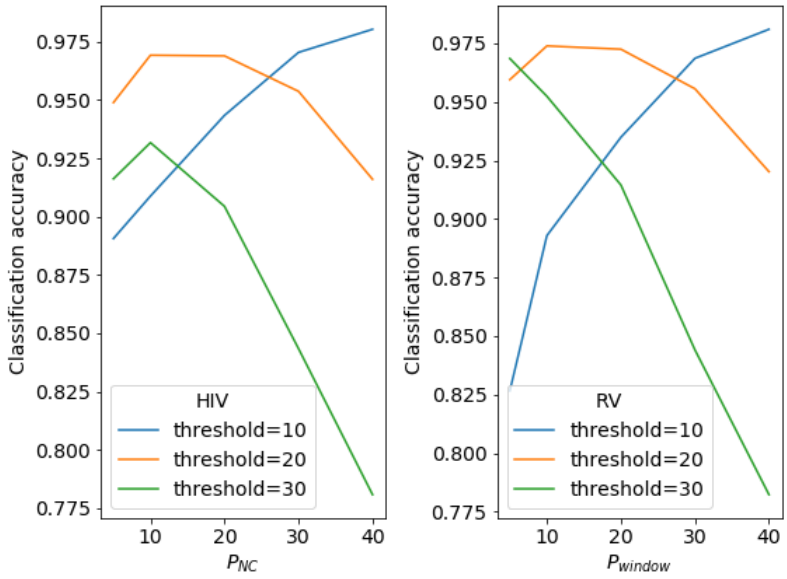


Figure 6.6: The classification accuracies under different settings. The left subplot shows the results when the aberrant scores are calculated as the variance of the high index normalized coefficients under different values of P_{NC} and the right subplot shows the results when the aberrant scores are calculated as the variance of the rolling variance under different values of P_{window}. The prediction accuracies for the largest absolute value method are 91.45%, 85.65% and 68.68% when $threshold = 10, 20, 30$.

For the FPR and Fence approaches, only the normal training images are used to build the aberrant data detector and to determine *threshold*. Figures 6.7 and 6.8 show the results when using the FPR and Fence to determine the threshold. We observe that the lower the fpr is, the higher the threshold we need to set, and thus the higher the accuracy we would obtain in detecting the normal images. Similarly, for a larger value of M in the Fence approach, we would set a larger value for the fence, which results in greater accuracy in classifying the normal images. The classification

	threshold			*fpr*			*M*		
	10	20	30	0.05	0.01	0.005	2	3	4
P_{NC}	40	10	10	10	5	20	20	5	10
P_{window}	40	10	5	10	5	10	20	5	40

Table 6.1: The selected values of P_{NC} and P_{window} under different settings.

accuracy is not sensitive to the change in the values of P_{NC} or P_{window}. We selected the values of P_{NC} and P_{window} by CV, which are shown in Table 6.1.

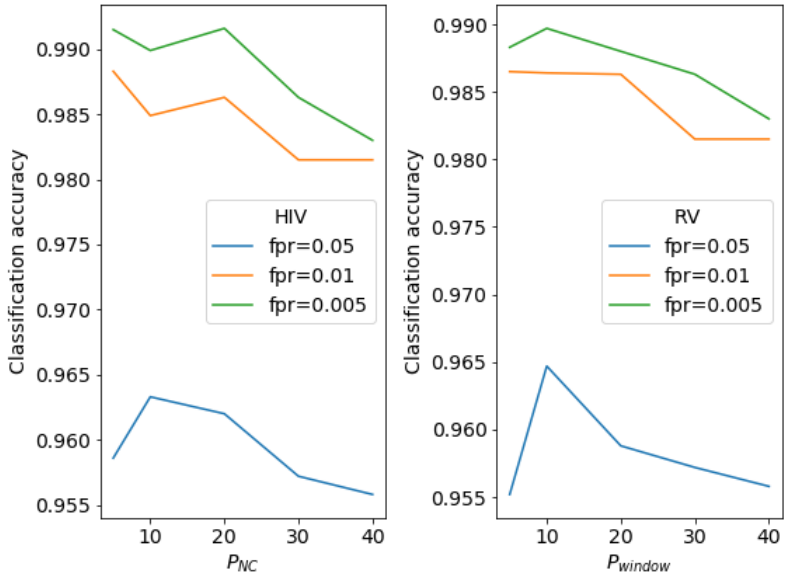


Figure 6.7: The classification accuracies for different fpr under different settings. The left subplot shows the results when the aberrant scores are calculated as the variance of the high index normalized coefficients under different values of P_{NC} and the right subplot shows the results when the aberrant scores are calculated as the variance of the rolling variance under different values of P_{window}. The prediction accuracies for the largest absolute method are 94.47%, 99.51% and 99.67% when the fpr is 0.05, 0.01, 0.005, respectively.

Receiver Operating Characteristic (ROC) curve for the basic detector

To evaluate the overall detection performances, we calculate the Receiver Operating Characteristic (ROC) curve for different approaches to calculating the aberrant score (HIV, RV and MAV). The results are shown in Figure 6.9, with the area under curve

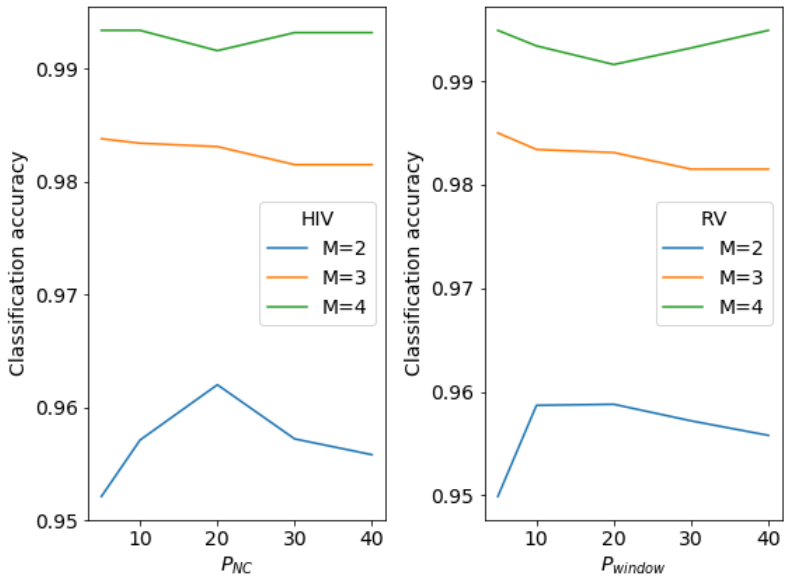


Figure 6.8: The classification accuracies for different multiplier, M, of the fence under different settings. The left subplot shows the results when the aberrant scores are calculated as the variance of the high index normalized coefficients under different values of P_{NC} and the right subplot shows the results when the aberrant scores are calculated as the variance of the rolling variance under different values of P_{window}. The prediction accuracies for the largest absolute method are 95.27%, 98.82% and 99.85% when the multipliers, M, is 2, 3, 4, respectively.

(AUC) equaling 0.9760, 0.9755 and 0.9697 for the three methods, respectively. We observe that the AUCs for HIV and RV are almost the same. The AUC for MAV is slightly smaller, but the results are still satisfactory. The ROC curves suggest that the basic detectors built upon these three methods would all be powerful tools to distinguish normal images from abnormal images.

Classification results for aberrant data detector

Instead of classifying images into two categories (normal v.s. abnormal), we can further divide the abnormal images into adversarial v.s. corrupted. The mechanism is basically the same except that if a query sample is identified as abnormal, we would perform spike-replacement and classify the data point again. Through this extra step, we are capable of classifying the testing data into three categories. The testing dataset

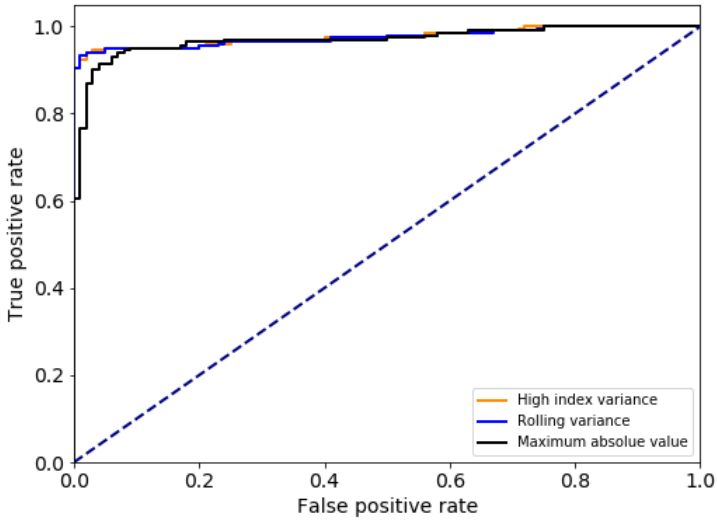


Figure 6.9: The ROC curve of the aberrant data detector. The area under curve is 0.9760, 0.9755 and 0.9697 for the method HIV, RV and MAV, respectively.

contains 100 normal images, 100 corrupted images and 100 adversarial images. We adopt the settings in Table 6.1 using different aberrant scores calculations and given thresholds. Our studies show that the performance is not sensitive to *cutoff*. Thus, we set $cutoff = 30$. We calculate the detection accuracy (Acc.), sensitivity (Sen.) and specificity (Spec.) for each type of image. The sensitivity and the specificity are calculated using the so-called "one v.s. other" approach. As an example, the true positives of the normal images are all normal images that are detected as normal; the false positives of the normal images are all images that are abnormal (could be corrupted or adversarial) but are detected as normal. Similar calculations are applied to the corrupted and adversarial images. The results of the aberrant data detector adopting different methods are shown in Tables 6.2, 6.3 and 6.4. MAV may not be suitable to calculate the aberrant score, since it can hardly catch the variability. The sensitivity to the corrupted images is low when we use small fpr or large M. Satisfactory detection rates are obtained with suitably chosen tuning parameters.

		$threshold = 10$			$threshold = 20$			$threshold = 30$		
		Acc.	Sen.	Spec.	Acc.	Sen.	Spec.	Acc.	Sen.	Spec.
HIV	Normal	0.95	0.98	0.94	0.95	0.97	0.94	0.92	0.99	0.89
	Corrupted	0.98	0.99	0.97	0.97	0.96	0.97	0.95	0.87	0.99
	Adversarial	0.95	0.86	1.00	0.96	0.88	0.99	0.96	0.89	1.00
RV	Normal	0.95	0.98	0.94	0.95	0.97	0.94	0.95	0.99	0.93
	Corrupted	0.98	0.99	0.97	0.97	0.96	0.97	0.95	0.87	0.99
	Adversarial	0.95	0.86	1.00	0.95	0.88	0.98	0.93	0.89	0.95
MAV	Normal	0.91	0.83	0.95	0.86	0.98	0.80	0.73	0.99	0.60
	Corrupted	0.98	0.99	0.97	0.89	0.69	0.99	0.77	0.31	1.00
	Adversarial	0.91	0.88	0.93	0.95	0.88	0.99	0.96	0.90	0.99

Table 6.2: The performance of the aberrant data detector when specifying *threshold* directly.

		$fpr = 0.05$			$fpr = 0.01$			$fpr = 0.005$		
		Acc.	Sen.	Spec.	Acc.	Sen.	Spec.	Acc.	Sen.	Spec.
HIV	Normal	0.90	1.00	0.85	0.82	1.00	0.73	0.83	1.00	0.75
	Corrupted	0.94	0.83	1.00	0.86	0.58	1.00	0.87	0.61	1.00
	Adversarial	0.96	0.88	1.00	0.96	0.88	0.99	0.95	0.87	0.99
RV	Normal	0.91	1.00	0.86	0.84	1.00	0.77	0.83	1.00	0.75
	Corrupted	0.93	0.80	1.00	0.83	0.50	1.00	0.86	0.59	1.00
	Adversarial	0.94	0.88	0.97	0.91	0.88	0.93	0.94	0.87	0.98
MAV	Normal	0.89	0.98	0.85	0.81	0.99	0.71	0.80	0.99	0.70
	Corrupted	0.91	0.80	0.97	0.82	0.53	0.97	0.84	0.51	1.00
	Adversarial	0.94	0.84	0.99	0.94	0.84	0.99	0.96	0.90	0.99

Table 6.3: The performance of the aberrant data detector when specifying *threshold* using the FPR approach.

		$M = 2$			$M = 3$			$M = 4$		
		Acc.	Sen.	Spec.	Acc.	Sen.	Spec.	Acc.	Sen.	Spec.
HIV	Normal	0.91	1.00	0.87	0.82	1.00	0.73	0.83	1.00	0.74
	Corrupted	0.95	0.86	1.00	0.86	0.59	1.00	0.86	0.57	1.00
	Adversarial	0.96	0.88	1.00	0.96	0.88	1.00	0.93	0.86	0.97
RV	Normal	0.91	1.00	0.87	0.84	1.00	0.77	0.83	1.00	0.74
	Corrupted	0.95	0.86	1.00	0.84	0.52	1.00	0.86	0.57	1.00
	Adversarial	0.96	0.88	1.00	0.92	0.88	0.94	0.93	0.86	0.97
MAV	Normal	0.87	0.98	0.81	0.80	0.99	0.71	0.76	0.99	0.65
	Corrupted	0.91	0.73	1.00	0.83	0.52	0.98	0.76	0.29	1.00
	Adversarial	0.96	0.90	0.99	0.95	0.86	0.99	0.93	0.90	0.94

Table 6.4: The performance of the aberrant data detector when specifying *threshold* using the Fence approach.

6.2.4 Prediction results

Our proposed preprocessing system produces "corrected" images that can be used by other prediction methods. A testing image is passed through the preprocessing system before applying the chosen prediction method. Depending on the identified testing data type, the preprocessing method would apply different mechanisms to the testing data. One can then apply the selected prediction method(s) to the processed data. These methods need not be GLLiM. We use the following prediction methods to investigate the efficacies of the preprocessing system.

1. GLLiM: We use the GLLiM forward model for prediction. The GLLiM model is trained under $K = 20$, $L_w = 9$.

2. FGAM: Considering the predictor X and the scalar response Y, the functional generalized additive model (McLean et al., 2014) builds the relationship between X and Y as:

$$g(\mathbb{E}[Y|X]) = \beta_0 + \int F(X(t), t)dt, \tag{6.6}$$

   where β_0 is the intercept, g is a known link function and F is an unspecified smooth function to be estimated. In our case, we set $g(x) = x$ and let t be the index of the image pixel. The function $F(\cdot, \cdot)$ is estimated through tensor-product B-splines with roughness penalties. FGAM is built using the R package *refund* (Goldsmith et al., 2018). We build the model on each dimension of Y using 100 knots, which leads to three FGAM models. We use the default values for the rest of the settings.

3. SAM: Similar to lasso (Tibshirani, 1996), the Sparse additive model (Ravikumar et al., 2009) introduces the L_1 penalty to encourage sparse solutions on the functional coefficients. For the predictor X and the scalar response Y, SAM

aims to find the solution that minimizes

$$\mathbb{E}\left\{Y - \sum_{d=1}^{D} \beta_d f_d(X_d)\right\}^2 \tag{6.7}$$

subject to

$$\sum_{d=1}^{D} |\beta_d| \leq P \tag{6.8}$$

$$\mathbb{E}[f_d^2] = 1, \tag{6.9}$$

where f_d is a function to be estimated, $\beta = (\beta_1, ..., \beta_D)^\top$ is a vector and P is a scalar constraint. The constraint of β imposes the sparsity of the estimated β. The model is trained using the R package *SAM* (Zhao et al., 2014) under the default setting. We build a predictive model for each dimension of Y separately.

The prediction mean squared errors (PMSE) of normal, adversarial, corrupted and reconstructed testing datasets are shown in Table 6.5. The testing datasets contain 100 testing images. The reconstructed testing datasets show the outcomes of the data reconstruction on the corrupted datasets. Note that the adversarial dataset is generated against the GLLiM forward model. However, the prediction loss is still large when the other two predictive methods are used, which implies the transferability of the adversarial examples (Papernot et al., 2016). The improvements on the reconstructed datasets demonstrate the benefits of adopting the preprocessing system. With the data reconstruction procedure, we can effectively reduce the prediction errors on the corrupted images.

To evaluate the performances under different combinations of the normal and abnormal images, we conduct numerical experiments on different testing datasets as follows:

- Case A: The testing datasets contain 100 normal images and 100 corrupted

	Normal	Adversarial	Corrupted	Reconstructed
GLLiM	0.0199	0.2625	0.3189	0.0437
FGAM	0.4382	0.5454	2.6465	0.4756
SAM	0.1201	0.3176	0.3899	0.1374

Table 6.5: The PMSE of different types of images using different prediction models.

images.

- Case B: The testing datasets contain 100 normal images and 100 adversarial images.

- Case C: The testing datasets contain 100 normal images, 100 corrupted images and 100 adversarial images. The normal images are doubled weighted when calculating the MSE.

Table 6.6 shows the prediction results with ($threshold = 10$, $fpr = 0.05$, $M = 2$) and without (Baseline) the preprocessing system. In Table 6.6 we summarize the results when the aberrant scores are calculated using HIV. For the complete results using different settings of aberrant scores and classification thresholds, please see Appendix C. By comparing the PMSE under Baseline, GLLiM shows its superiority against other methods for modeling high-to-low non-linear associations. For each prediction method, the preprocessing system effectively reduces the prediction errors, which shows that the proposed system is a general approach to robustly handle normal and abnormal testing entries. We obtain the best prediction performance using the DA approach. However, it may be time-consuming to identify a suitable *threshold* using this approach since the appropriate value for *threshold* may vary from dataset to dataset. On the other hand, using the FPR and $Fence$ methods is more straightforward. We suggest starting with the FPR or $Fence$ method and using the resulting *threshold* as a starting point in the follow-up tuning process.

Pred. method	Test Case	Baseline	$threshold = 10$	$fpr = 0.05$	$M = 2$
GLLiM	A	0.1694	0.0312	0.0454	0.0415
	B	0.1412	0.0707	0.0743	0.0743
	C	0.1553	0.0509	0.0598	0.0579
FGAM	A	0.3899	0.1260	0.1373	0.1331
	B	0.2188	0.1559	0.1585	0.1585
	C	0.2369	0.1410	0.1479	0.1458
SAM	A	2.6465	0.3893	0.5035	0.4407
	B	0.491823	0.4700	0.4686	0.4686
	C	1.017092	0.4297	0.4861	0.4547

Table 6.6: The prediction mean squared errors (PMSE) under different experimental settings. The Baseline column shows the original PMSE. The rest of the columns present the PMSE using different classification thresholds when the aberrant scores are calculated using HIV.

6.3 Conclusion

The proposed preprocessing system shows its ability to improve prediction performance. The system contains two building blocks: an aberrant data detector and an aberrant data corrector. The aberrant data detector classifies the testing data into three categories: normal, corrupted and adversarial. Depending on the identified category, the aberrant data corrector applies different correction mechanisms. The data reconstruction process is devised for corrupted data. Using the inverse regression learned by GLLiM, we can reconstruct the damaged data effectively. Nearest neighbors are used to replace adversarial samples. Using three existing predictive models as examples, we demonstrate the generality of the system and elucidate the necessity of the system for obtaining reliable prediction outcomes.

CHAPTERS VII

Conclusion and Future Work

Predicting with regression models is widely used in many fields such as decision making, disease diagnosis and marketing, among others. It is therefore of crucial importance to be able to obtain reliable prediction outcomes. This dissertation focused on improving prediction efficacy in both the training and the testing stages. In particular, two robust training methods were devised, which can be accelerated using the proposed parallelization framework when dealing with large-scale datasets. To evaluate the model's robustness, we developed an efficient approach to generating adversarial examples. A preprocessing method was then established, which can handle different types of abnormal testing inputs. Experimental results on both synthetic and real-world datasets validated the enhancement of prediction performance in both the training and the testing stage.

In Chapter II, we analyzed in depth Gaussian Locally Linear Mapping (GLLiM) and proposed a method to reduce its prediction errors. Specifically, GLLiM adopts a clustering-based approach to approximate non-linear mappings. However, these clusters can contain sub-clusters or singletons, which can dramatically downgrade the prediction performance. We established Hierarchical Gaussian Locally Linear Mapping (HGLLiM), which follows a two-layer hierarchical clustering structure to accommodate sub-clusters. A robust estimation procedure was devised to remove

outliers, which results in improved model stability. Numerical results demonstrated that HGLLiM could construct models that are robust against both complex data structures and outliers.

In Chapter III, we extended HGLLiM with a parallelization framework that can substantially accelerate HGLLiM's training process. By separating data into distinct groups, we can train sub-models in a parallelized fashion. The hierarchical structure enables straightforward model aggregation. Sub-models can be integrated into the final model by adding an extra latent variable for data groups. The acceleration technique for HGLLiM was applied to two real-world applications. For the fingerprint dataset, we can achieve a 75% reduction in computational time compared to the previous method, while maintaining comparable prediction performance on one microvascular variable, blood volume fraction. Furthermore, the prediction errors on apparent diffusion coefficient were much smaller than those produced by the previous method. Another numerical investigation on the sound dataset demonstrated that parallelized HGLLiM could reduce around 70% of the training time compared to a neural-network-based approach with similar prediction performance.

In Chapter IV, we proposed Robust Gaussian Locally Linear Mapping (RGLLiM), which inherits the advantages of GLLiM for tackling high-dimensionality and non-linearity. For the concern of model stability, RGLLiM further removes outliers and limits relative cluster sizes. Experimental results showed the capability of RGLLiM to provide reliable prediction outcomes.

In Chapter V, we devised the Autoencoder-based Zeroth Order Optimization Method (AutoZOOM) to efficiently generate adversarial examples under the black-box setting. The adversarial examples are essential for evaluating model robustness against malicious inputs. AutoZOOM adopts gradient vector estimation and dimension reduction techniques to improve the efficiency of the generation process. The experimental results demonstrated that AutoZOOM can generate adversarial exam-

ples toward state-of-the-art deep neural networks successfully and efficiently.

In Chapter VI, we established a preprocessing method to enhance prediction performances by carefully handling testing inputs. Data types are determined by the aberrant data detector. Based on the detection results, different actions are taken by the aberrant data corrector. For reconstructing corrupted data, we utilized the inverse associations obtained from GLLiM and showed its capability of repairing damaged images. As for adversarial samples, nearest neighbors are used as surrogates. We demonstrated the high detection performance of the aberrant data detector for distinguishing different types of data. Through the preprocessing method, users have the flexibility of selecting their preferred predictive methods. Results from three predictive models illustrated that the proposed preprocessing approach can substantially improve the prediction performance for abnormal testing data.

There are many interesting directions worth further investigation. At the training stage, we achieve model robustness by removing outliers. A possible alternative is to estimate model parameters robustly without removing any training samples. Perthame et al. (2018) achieved this goal through robust clustering. A similar idea can be applied when estimating the regression parameters. Local linearity is assumed to tackle the non-linear issue. To relax this restriction, one can apply kernel methods when learning the inverse associations within clusters.

Adversarial generation can be further improved by adopting other methods such as the attention technique to find a more representative latent space. In our settings, all of the output scores are known to the attackers. The work can be further extended if a more restrictive setting is considered. For example, we may consider the black-box setting when the attackers only have knowledge of the top-n prediction class or even only the decision (top-1) of the model.

The robust system can be extended to a more general scenario in which the training dataset includes abnormal observations. These abnormal data would provide in-

correct information when training the aberrant data detector and should be removed. We could consider different kinds of abnormal data such as data with different kinds of additive noise. Building a system that is robust against different kinds of abnormal data could be an interesting and challenging problem.

APPENDICES

APPENDIX A

Appendix of Chapter IV

A.1 Details for the E-step

For a given cluster, $Z = k$, the associations among X, T and W can be expressed as $X = A_k^t T + A_k^w W + b_k + E_k$, where $T \sim \mathcal{N}(c_k^t, \Gamma_k^t)$, $W \sim \mathcal{N}(c_k^w, \Gamma_k^w)$, $E_k \sim \mathcal{N}(0, \Sigma_k)$. Under the assumption that T and W are independent, we can write the joint density as:

$$\begin{bmatrix} T \\ W \\ X \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} c_k^t \\ c_k^w \\ A_k^t c_k^t + A_k^w c_k^w + b_k \end{bmatrix}, \begin{bmatrix} \Gamma_k^t & 0 & \Gamma_k^t A_k^{t\top} \\ 0 & \Gamma_k^w & \Gamma_k^w A_k^{w\top} \\ A_k^t \Gamma_k^t & A_k^w \Gamma_k^w & \Sigma_k + A_k^t \Gamma_k^t A_k^{t\top} + A_k^w \Gamma_k^w A_k^{w\top} \end{bmatrix} \right)$$

Consider

$$
\begin{aligned}
&\mathbb{E}[p(y_n|x_n, Z_n = k; \theta)] \\
&= \begin{bmatrix} c_k^t \\ c_k^w \end{bmatrix} + \begin{bmatrix} \Gamma_k^t & 0 \\ 0 & \Gamma_k^w \end{bmatrix} \begin{bmatrix} A_k^{t\top} \\ A_k^{w\top} \end{bmatrix} (\Sigma_k + A_k^t \Gamma_k^t A_k^{t\top} + A_k^w \Gamma_k^w \Gamma_k^{w\top})^{-1} \left(x_n - b_k - \begin{bmatrix} A_k^t & A_k^w \end{bmatrix} \begin{bmatrix} c_k^t \\ c_k^w \end{bmatrix} \right) \\
&= \begin{bmatrix} c_k^t \\ c_k^w \end{bmatrix} + \begin{bmatrix} \Gamma_k^t A_k^{t\top} \\ \Gamma_k^w A_k^{w\top} \end{bmatrix} (\Sigma_k + A_k^t \Gamma_k^t A_k^{t\top} + A_k^w \Gamma_k^w \Gamma_k^{w\top})^{-1} (x_n - A_k^t c_k^t - A_k^w c_k^w - b_k) \\
&= \begin{bmatrix} c_k^t \\ c_k^w \end{bmatrix} + \begin{bmatrix} \Gamma_k^t A_k^{t\top} \\ \Gamma_k^w A_k^{w\top} \end{bmatrix} (\Gamma_k^*)^{-1} (x_n - A_k^t c_k^t - A_k^w c_k^w - b_k).
\end{aligned}
$$

Since T and W are independent, we have

$$\mathbb{E}[p(t_n|x_n, Z_n = k; \theta)] = c_k^t + \Gamma_k^t A_k^{t\top} (\Gamma_k^*)^{-1} (x_n - A_k^t c_k^t - A_k^w c_k^w - b_k).$$

Next, we replace Γ_k^* with its constraint version to obtain $\tilde{\mu}_k$.

A.2 Finding truncated eigenvalues

We adopt the key results in Fritz et al. (2013) to find m such that Equation (4.21) is minimized. Consider $e_1 \leq e_2 \leq \cdots \leq e_{2KD}$ obtained by ordering the following values:

$$\lambda_1(S_1), ..., \lambda_d(S_k), ..., \lambda_D(S_K), \lambda_1(S_1)/C, ..., \lambda_d(S_k)/C, ..., \lambda_D(S_K)/C.$$

Also, consider $f_1, ..., f_{2KD+1}$ such that $f_1 \leq e_1 \leq f_2 \leq e_2 \leq \cdots \leq f_{2KD} \leq e_{2KD} \leq f_{2KD+1}$. For each f_i, $i = 1, ..., 2KD+1$, the corresponding m_i is

$$m_i = \frac{\sum_{k=1}^{K} N_k \left(\sum_{d=1}^{D} \lambda_{k,d} \mathbb{I}(\lambda_{k,d} < f_i) + \frac{1}{C} \sum_{d=1}^{D} \lambda_{k,d} \mathbb{I}(\lambda_{k,d} > C f_i) \right)}{\sum_{k=1}^{K} N_k \left(\sum_{d=1}^{D} \mathbb{I}(\lambda_{k,d} < f_i) + \mathbb{I}(\lambda_{k,d} > C f_i) \right)} \tag{A.1}$$

where $\mathbb{I}$ is the indicator function and $N_k = \sum_{n=1}^{N} r_{nk}$. We choose the m that minimizes Equation (4.21), and the corresponding $\tilde{\Sigma}_k$ for all k are positive definite. There is a chance that no m can result in all valid $\tilde{\Sigma}_k$. If this happens, we estimate a lower bound for m. The covariance matrix of Y (Γ_k^*) is the sum of two terms, $A_k^t \Gamma_k^t A_k^{t\top}$, from T, and $(A_k^w \Gamma_k^w A_k^{w\top} + \Sigma_k)$, variances from latent W and the random errors. To make $\tilde{\Sigma}_k$ a valid covariance matrix, Γ_k^* cannot be smaller than $A_k^t \Gamma_k^t A_k^{t\top}$. That is $\Gamma_k^* - A_k^t \Gamma_k^t A_k^{t\top}$ should be positive definite. We can rewrite $\Gamma_k^* - A_k^t \Gamma_k^t A_k^{t\top} = \Gamma_k^* \left(\mathbf{I}_{L_t} - (\Gamma_k^*)^{-1} A_k^t \Gamma_k^t A_k^{t\top} \right)$. Thus, the largest eigenvalue of $(\Gamma_k^*)^{-1} A_k^t \Gamma_k^t A_k^{t\top}$ needs to be less than 1. Denote $\lambda_{max}(A)$ as the largest eigenvalue of matrix A. We have $\lambda_{max}((\Gamma_k^*)^{-1} A_k^t \Gamma_k^t A_k^{t\top}) < 1$. As given in Equation (4.22), the largest eigenvalue of $(\Gamma_k^*)^{-1}$ is $1/m$. Thus, we choose $m > \lambda_{max}(A_k^t \Gamma_k^t A_k^{t\top})$. This is the lower bound of m which makes $\tilde{\Sigma}_k$ a valid covariance matrix. If no suitable m is obtainable, we choose $m = \max_{k=1...K} \lambda_{max}(A_k^t \Gamma_k^t A_k^{t\top})$.

A.3 Simulating clustered data

We generate simulated data with a clustered structure based on the orange juice dataset. We first identify the "core" members of the clusters and use this information for the simulation.

Using GLLiM, we conduct a 100-fold cross-validation experiment with GLLiM ($K = 10, L_w = 10$) and count the times that two data points are clustered together. For each experiment, GLLiM is randomly initialized 10 times, and the initial values with the largest likelihood are used. Figure A.1(a) shows the adjacency matrix of the

counts after permutation. We pick selected groups so that some groups are completely different, such as Group 1 and Group 4, and some overlap, such as Group 2, and Group 3.

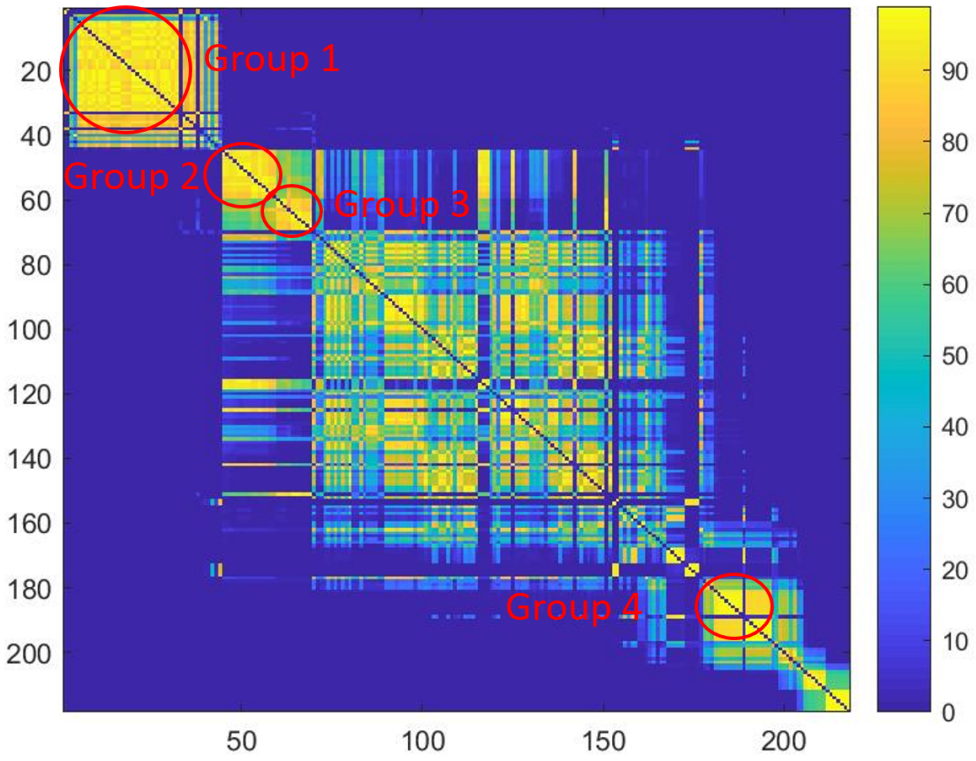


Figure A.1: The adjacency matrix of GLLiM clusters after permutation and the identified sub-groups.

Once we identify the core members of different clusters, we simulate data using the group information obtained from these data points. Denote data coming from the cluster k as $(t_n, y_n)_{n=1}^{N}$. The estimated parameters are obtained using the following procedure.

1. Estimate the low-dimensional cluster parameter:

$$c_k^t = \frac{1}{N}\sum_{n=1}^{N} t_n,$$
$$\Gamma_k^t = \frac{1}{N}\sum_{n=1}^{N}(t_n - c_k^t)(t_n - c_k^t)^\top.$$

2. Estimate the regression parameters

$$A_k^t = \bar{Y}\bar{T}^\top(\bar{T}\bar{T}^\top)^{-1},$$
$$b_k = \frac{1}{N}\sum_{n=1}^{N}(y_n - A_k^t t_n),$$

where

$$\bar{T} = [(t_1 - \bar{t}), ..., (t_N - \bar{t})],$$
$$\bar{Y} = [(y_1 - \bar{y}), ..., (y_N - \bar{y})],$$
$$\bar{t} = \frac{1}{N}\sum_{n=1}^{N} t_n,$$
$$\bar{y} = \frac{1}{N}\sum_{n=1}^{N} y_n.$$

3. The next step is to estimate A_k^w and Σ_k. Defining the residual as $u_{nk} = (y_n - A_k^t t_n - b_k)$, the optimal values for A_k^w and Σ_k should minimize the following criterion as in Deleforge et al. (2015):

$$Q_k(A_k^w, \Sigma_k) = -\frac{1}{2}\sum_{n=1}^{N}\left(\log|\Sigma_k + A_k^w A_k^{w\top}| + u_{nk}^\top(\Sigma_k + A_k^w A_k^{w\top})^{-1}u_{nk}\right).$$

With the assumption that $\Sigma_k = \sigma_k^2 I_D$, we consider two ways of estimating A_k^w and Σ_k. The first one adopts the key results in PPCA. Let $R_k = \frac{1}{N}\sum_{n=1}^{N} u_{nk}u_{nk}^\top$

be the covariance matrix of the residuals. We obtain

$$A_k^w = U_k(\Lambda_k - \sigma_k^2 I_{L_w})^{1/2},$$
$$\sigma_k^2 = \frac{\sum_{d=L_w+1}^{D} \lambda_{dk}}{D - L_w},$$

where U_k is the matrix containing the first L_w eigenvectors of R_k and λ_{dk} denotes the eigenvalues of R_k with $\lambda_{1k} > \cdots > \lambda_{Dk}$.

An alternative approach is to estimate the eigenfunctions using Functional Principal Component Analysis (FPCA). We use the data in Group 1 for illustration. Figure A.2 shows the first two eigenvectors of the residual covariance matrix obtained from the classical eigenvalue decomposition and their FPCA counterparts. We use the software provided in Chen and Lei (2015) and set $\rho_1 = 0.7985$, which is selected through cross-validation, to obtain smooth eigenfunctions. In addition, we choose $L_w = 8$ to capture the dependence between covariates.

4. To simulate data for a chosen group, we first generate the observed t'_n from the distribution $\mathcal{N}(c_k^t, \Gamma_k^t)$ and the latent w'_n from $\mathcal{N}(0, I_{L_w})$. We let the low-dimensional data be $x'_n = [t'_n; w'_n]$; the corresponding high-dimensional y'_n is generated from distribution $\mathcal{N}(A_k x'_n + b_k, \Sigma_k)$. Figure A.3 shows the data simulated from Groups 1–4.

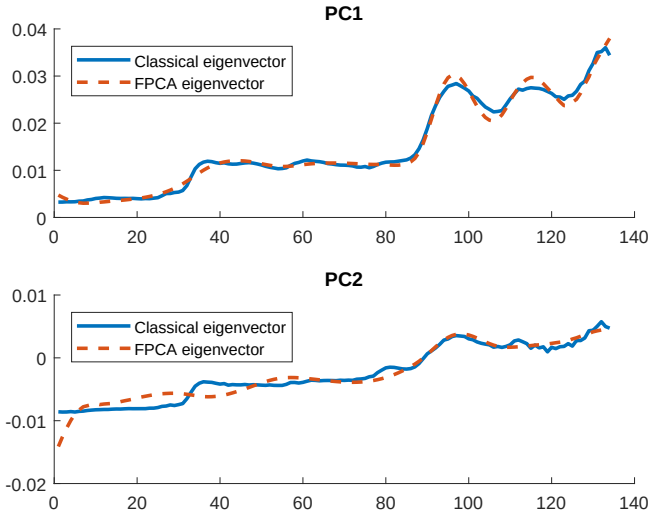


Figure A.2: The first two principal components and their counterparts obtained from LFPCA.

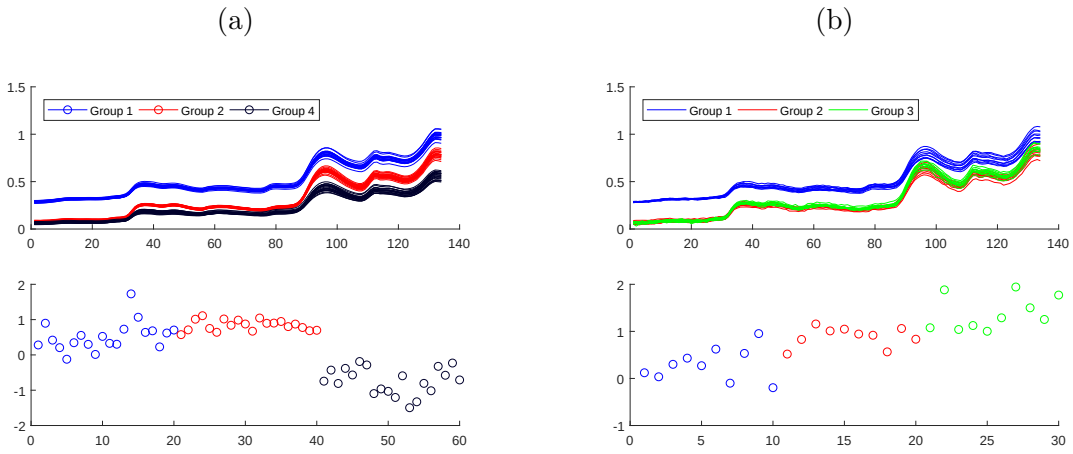


Figure A.3: The simulated orange juice data with (a) distinct clusters and (b) overlapped clusters.

APPENDIX B

Appendix of Chapter V

B.1 Proof of Theorem 5.1

Recall that the data dimension is D and we assume f to be differentiable and its gradient ∇f to be L-Lipschitz. Fixing β, consider a smoothed version of f:

$$f_\beta(x) = \mathbb{E}_u[f(x + \beta u)]. \tag{B.1}$$

Based on Gao et al. (2014), Lemma 4.1-a, we have the relation

$$\nabla f_\beta(x) = \mathbb{E}_u \left[\frac{D}{\beta} f(x + \beta u) u \right] = \frac{D}{b} \mathbb{E}_u \left[g \right], \tag{B.2}$$

which then yields

$$\mathbb{E}_u \left[g \right] = \frac{b}{D} \nabla f_\beta(x), \tag{B.3}$$

where we recall that g has been defined in Equation (5.7). Moreover, based on Gao et al. (2014), Lemma 4.1-a, we have

$$\|\nabla f_\beta(x) - \nabla f(x)\|_2 \leq \frac{\beta D L}{2}. \tag{B.4}$$

Substituting (B.3) into (B.4), we obtain

$$\|\mathbb{E}[g] - \frac{b}{D}\nabla f(x)\|_2 \leq \frac{\beta b L}{2}.$$

This then implies that

$$\mathbb{E}[g] = \frac{b}{D}\nabla f(x) + \boldsymbol{\epsilon}, \tag{B.5}$$

where $\|\boldsymbol{\epsilon}\|_2 \leq \frac{b\beta L}{2}$.

Once again, by applying Gao et al. (2014), Lemma 4.1-b, we can easily obtain that

$$\mathbb{E}_u[\|g\|_2^2] \leq \frac{b^2 L^2 \beta^2}{2} + \frac{2b^2}{D}\|\nabla f(x)\|_2^2. \tag{B.6}$$

Now, let us consider the averaged random gradient estimator in Equation (5.8),

$$\overline{g} = \frac{1}{q}\sum_{i=1}^{q} g_i = \frac{b}{q}\sum_{i=1}^{q} \frac{f(x+\beta u_i) - f(x)}{\beta} u_i.$$

Due to the properties of i.i.d. samples $\{u_i\}$ and (B.5), we define

$$v =: \mathbb{E}[g_i] = \frac{b}{D}\nabla f(x) + \boldsymbol{\epsilon}. \tag{B.7}$$

Moreover, we have

$$\begin{aligned}
\mathbb{E}[\|\overline{g}\|_2^2] =& \mathbb{E}\left[\left\|\frac{1}{q}\sum_{i=1}^{q}(g_i - v) + v\right\|_2^2\right] && \text{(B.8)}\\
=& \|v\|_2^2 + \mathbb{E}\left[\left\|\frac{1}{q}\sum_{i=1}^{q}(g_i - v)\right\|_2^2\right]\\
=& \|v\|_2^2 + \frac{1}{q}\mathbb{E}[\|g_1 - v\|_2^2] && \text{(B.9)}\\
=& \|v\|_2^2 + \frac{1}{q}\mathbb{E}[\|g_1\|_2^2] - \frac{1}{q}\|v\|_2^2, && \text{(B.10)}
\end{aligned}$$

where we have used the fact that $\mathbb{E}[g_i] = \mathbb{E}[g_1] = v \ \forall \ i$. The definition of v in (B.7) yields

$$\begin{aligned}
\|v\|_2^2 \leq& 2\frac{b^2}{D^2}\|\nabla f(x)\|_2^2 + 2\|\boldsymbol{\epsilon}\|_2^2\\
\leq& 2\frac{b^2}{D^2}\|\nabla f(x)\|_2^2 + \frac{1}{2}b^2\beta^2L^2. && \text{(B.11)}
\end{aligned}$$

From (B.6), we also obtain that for any i,

$$\mathbb{E}[\|g_i\|_2^2] \leq \frac{b^2L^2\beta^2}{2} + \frac{2b^2}{D}\|\nabla f(x)\|_2^2. \tag{B.12}$$

Substituting (B.11) and (B.12) into (B.10), we obtain

$$\begin{aligned}
\mathbb{E}[\|\overline{g}\|_2^2] \leq& \|v\|_2^2 + \frac{1}{q}\mathbb{E}[\|g_1\|_2^2] && \text{(B.13)}\\
\leq& 2(\frac{b^2}{D^2} + \frac{b^2}{Dq})\|\nabla f(x)\|_2^2 + \frac{q+1}{2q}b^2L^2\beta^2. && \text{(B.14)}
\end{aligned}$$

Finally, we bound the mean squared estimation error as

$$\begin{aligned}\mathbb{E}[\|\overline{g} - \nabla f(x)\|_2^2] &\leq 2\mathbb{E}[\|\overline{g} - v\|_2^2] + 2\|v - \nabla f(x)\|_2^2 \\ &\leq 2\mathbb{E}[\|\overline{g}\|_2^2] + 2\|\frac{b}{D}\nabla f(x) + \boldsymbol{\epsilon} - \nabla f(x)\|_2^2 \\ &\leq 4(\frac{b^2}{D^2} + \frac{b^2}{Dq} + \frac{(b-D)^2}{D^2})\|\nabla f(x)\|_2^2 \\ &+ \frac{2q+1}{q} b^2 L^2 \beta^2, \end{aligned} \tag{B.15}$$

which completes the proof.

B.2 Architectures of convolutional autoencoder (CAE)

Dataset:	MNIST Training MSE: 2.00×10^{-3}
	Reduction ratio / image size / feature map size: 25% / 28×28×1 / 14×14×1
Encoder:	ConvReLU-16 → ConvReLU-16 → MaxPool → Conv-1
Decoder:	ConvReLU-16 → Reshape-Re-U → ConvReLU-16 → Conv-1

Dataset:	CIFAR-10 Training MSE: 5.00×10^{-3}
	Reduction ratio / image size / feature map size: 6.25% / 32×32×3 / 8×8×3
Encoder:	ConvReLU-16 → ConvReLU-16 → MaxPool → ConvReLU-3 → MaxPool → Conv-3
Decoder:	ConvReLU-16 → Reshape-Re-U → ConvReLU-16 → Reshape-Re-U → ConvReLU-16 → Conv-3

Dataset:	ImageNet Training MSE: 1.02×10^{-2}
	Reduction ratio / image size / feature map size: 1.15% / 299×299×3 / 32×32×3
Encoder:	Reshape-Bi-D → ConvReLU-16→ ConvReLU-16→ MaxPool → ConvReLU-3 → MaxPool → ConvReLU-3 → MaxPool → Conv-3
Decoder:	ConvReLU-16 → Reshape-Re-U → ConvReLU-16 → Reshape-Re-U → ConvReLU-16 → Reshape-Re-U → ConvReLU-16 → Reshape-Bi-U → Conv-3

ConvReLU-16: Convolution (16 filters, kernel size: 3×3×*Dep*) + ReLU activation
ConvReLU-3: Convolution (3 filters, kernel size: 3×3×*Dep*) + ReLU activation
Conv-3: Convolution (3 filters, kernel size: 3×3×*Dep*) Conv-1: Convolution (1 filter, kernel size: 3×3×*Dep*)
Reshape-Bi-D: Bilinear reshaping from 299×299×3 to 128×128×3
Reshape-Bi-U: Bilinear reshaping from 128×128×16 to 299×299×16
Reshape-Re-U: Reshaping by replicating pixels from $U \times V \times Dep$ to $2U \times 2V \times Dep$
Dep: a proper depth

Table B.1: Architectures of Autoencoders in AutoZOOM

APPENDIX C

Appendix of Chapter VI

C.1 Prediction performance under different settings

GLLiM

The PMSE when no preprocessing method is adopted for Cases A, B and C are 0.1694, 0.1412 and 0.1553 respectively.

		$threshold = 10$	$threshold = 20$	$threshold = 30$
HIV	Case A	0.031202	0.034754	0.041909
	Case B	0.070776	0.072501	0.072036
	Case C	0.050989	0.053628	0.056972
RV	Case A	0.031202	0.032552	0.037105
	Case B	0.070776	0.072501	0.072036
	Case C	0.050989	0.052526	0.054570
MAV	Case A	0.037509	0.050431	0.087487
	Case B	0.076682	0.075590	0.075001
	Case C	0.057095	0.063011	0.081244

Table C.1: The prediction performance of different testing cases when setting the classification threshold directly. The GLLiM forward model is used for conducting prediction.

		$fpr = 0.05$	$fpr = 0.01$	$fpr = 0.005$
HIV	Case A	0.045412	0.057492	0.061520
	Case B	0.074334	0.074334	0.074724
	Case C	0.059873	0.065913	0.068122
RV	Case A	0.046637	0.056961	0.061326
	Case B	0.074334	0.074334	0.074724
	Case C	0.060486	0.065648	0.068025
MAV	Case A	0.043972	0.068211	0.070342
	Case B	0.073700	0.072877	0.075001
	Case C	0.058836	0.070544	0.072672

Table C.2: The prediction performance of different testing cases when setting the classification threshold using the FPR method. The GLLiM forward model is used for conducting prediction.

		$M = 2$	$M = 3$	$M = 4$
HIV	Case A	0.041562	0.057526	0.059520
	Case B	0.074334	0.074334	0.076563
	Case C	0.057948	0.065930	0.068042
RV	Case A	0.041562	0.056070	0.059520
	Case B	0.074334	0.074334	0.076563
	Case C	0.057948	0.065202	0.068042
MAV	Case A	0.047114	0.070053	0.080792
	Case B	0.075850	0.073919	0.075001
	Case C	0.061482	0.071986	0.077896

Table C.3: The prediction performance of different testing cases when setting the classification threshold using the Fence approach. The GLLiM forward model is used for conducting prediction.

FGAM

The PMSE when no preprocessing method is adopted for Cases A, B and C are 2.646522, 0.491823 and 1.017092, respectively.

		$threshold = 10$	$threshold = 20$	$threshold = 30$
HIV	Case A	0.389335	0.394444	0.522734
	Case B	0.470091	0.461146	0.473503
	Case C	0.429713	0.427795	0.498119
RV	Case A	0.389335	0.387740	0.407265
	Case B	0.470091	0.461146	0.473503
	Case C	0.429713	0.424443	0.440384
MAV	Case A	0.381832	0.640236	1.090812
	Case B	0.459141	0.458783	0.459365
	Case C	0.420487	0.549509	0.775089

Table C.4: The prediction performance of different testing cases using FGAM when the preprocessing system is applied. The classification threshold of the preprocessing system is determined directly.

		$fpr = 0.05$	$fpr = 0.01$	$fpr = 0.005$
HIV	Case A	0.503558	0.787858	0.659772
	Case B	0.468684	0.468684	0.469046
	Case C	0.486121	0.628271	0.564409
RV	Case A	0.457181	0.659355	0.660893
	Case B	0.468684	0.468684	0.469046
	Case C	0.462932	0.564020	0.564969
MAV	Case A	0.536491	0.848910	0.891867
	Case B	0.457288	0.454621	0.459365
	Case C	0.496889	0.651765	0.675616

Table C.5: The prediction performance of different testing cases using FGAM when the preprocessing system is applied. The classification threshold of the preprocessing system is determined using the FPR approach.

		$M = 2$	$M = 3$	$M = 4$
HIV	Case A	0.440729	0.788169	0.647107
	Case B	0.468684	0.468684	0.471123
	Case C	0.454707	0.628427	0.559115
RV	Case A	0.440729	0.659014	0.647107
	Case B	0.468684	0.468684	0.471123
	Case C	0.454707	0.563849	0.559115
MAV	Case A	0.578610	0.857802	0.978572
	Case B	0.459592	0.456740	0.459365
	Case C	0.519101	0.657271	0.718969

Table C.6: The prediction performance of different testing cases using FGAM when the preprocessing system is applied. The classification threshold of the preprocessing system is determined using the Fence method.

SAM

The PMSE when no preprocessing method is adopted for Cases A, B and C are 0.3899, 0.2188 and 0.2369, respectively.

		$threshold = 10$	$threshold = 20$	$threshold = 30$
HIV	Case A	0.126032	0.131357	0.137214
	Case B	0.155991	0.155740	0.158995
	Case C	0.141011	0.143549	0.148105
RV	Case A	0.126032	0.131845	0.134953
	Case B	0.155991	0.155740	0.158995
	Case C	0.141011	0.143792	0.146974
MV	Case A	0.130786	0.158603	0.223830
	Case B	0.158590	0.155227	0.156772
	Case C	0.144688	0.156915	0.190301

Table C.7: The prediction performance of different testing cases using SAM when the preprocessing system is applied. The classification threshold of the preprocessing system is determined directly.

		$fpr = 0.05$	$fpr = 0.01$	$fpr = 0.005$
HIV	Case A	0.137384	0.177890	0.169134
	Case B	0.158538	0.158538	0.158631
	Case C	0.147961	0.168214	0.163882
RV	Case A	0.134064	0.171268	0.169590
	Case B	0.158538	0.158538	0.158631
	Case C	0.146301	0.164903	0.164111
MAV	Case A	0.141921	0.187817	0.194067
	Case B	0.155791	0.156024	0.156772
	Case C	0.148856	0.171921	0.175419

Table C.8: The prediction performance of different testing cases using SAM when the preprocessing system is applied. The classification threshold of the preprocessing system is determined using the FPR approach.

		$M = 2$	$M = 3$	$M = 4$
HIV	Case A	0.133141	0.178180	0.168970
	Case B	0.158538	0.158538	0.158750
	Case C	0.145840	0.168359	0.163860
RV	Case A	0.133141	0.168909	0.168970
	Case B	0.158538	0.158538	0.158750
	Case C	0.145840	0.163724	0.163860
MAV	Case A	0.152351	0.192249	0.217285
	Case B	0.157003	0.156512	0.156772
	Case C	0.154677	0.174380	0.187028

Table C.9: The prediction performance of different testing cases using SAM when the preprocessing system is applied. The classification threshold of the preprocessing system is determined using the Fence method.

BIBLIOGRAPHY

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283.

Allen, J. (1982). Applications of the short time fourier transform to speech processing and spectral analysis. In *International Conference on Acoustics, Speech, and Signal Processing*, Volume 7, pp. 1012–1015. IEEE.

Archambeau, C. and Verleysen, M. (2007). Robust Bayesian clustering. *Neural Networks 20*(1), 129–138.

Baek, J., McLachlan, G. J., and Flack, L. K. (2010). Mixtures of Factor Analyzers with Common Factor Loadings: Applications to the Clustering and Visualization of High-Dimensional Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*(7), 1298–1309.

Bai, X., Yao, W., and Boyer, J. E. (2012). Robust fitting of mixture regression models. *Computational Statistics & Data Analysis 56*(7), 2347–2359.

Banfield, J. D. and Raftery, A. E. (1993). Model-based Gaussian and non-Gaussian clustering. *Biometrics 49*, 803–821.

Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. (2009). PatchMatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics 28*(3), 24.

Bernardo, J., Bayarri, M., Berger, J., Dawid, A., Heckerman, D., Smith, A., and West, M. (2003). Bayesian factor regression models in the large p, small n paradigm. *Bayesian Statistics 7*, 733–742.

Bhagoji, A. N., Cullina, D., and Mittal, P. (2017). Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv preprint*.

Bioucas-Dias, J. M., Plaza, A., Dobigeon, N., Parente, M., Du, Q., Gader, P., and Chanussot, J. (2012). Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 5*(2), 354–379.

Bouveyron, C., Girard, S., and Schmid, C. (2007). High-dimensional data clustering. *Computational Statistics & Data Analysis 52*(1), 502–519.

Brown, M., Lewis, H. G., and Gunn, S. R. (2000). Linear spectral mixture models and support vector machines for remote sensing. *IEEE Transactions on Geoscience and Remote Sensing 38*(5), 2346–2360.

Brückner, M., Kanzow, C., and Scheffer, T. (2012). Static prediction games for adversarial learning problems. *Journal of Machine Learning Research 13*(Sep), 2617–2654.

Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pp. 39–57. IEEE.

Chamroukhi, F. (2016). Robust mixture of experts modeling using the t distribution. *Neural Networks 79*, 20–36.

Chang, W.-C. (1983). On using principal components before separating a mixture of two multivariate normal distributions. *Journal of the Royal Statistical Society: Series C (Applied Statistics) 32*, 267–275.

Chen, J. and Tan, X. (2009). Inference for multivariate normal mixtures. *Journal of Multivariate Analysis 100*(7), 1367–1383.

Chen, K. and Lei, J. (2015). Localized functional principal component analysis. *Journal of the American Statistical Association 110*(511), 1266–1275.

Chen, P.-Y., Sharma, Y., Zhang, H., Yi, J., and Hsieh, C.-J. (2018). EAD: elastic-net attacks to deep neural networks via adversarial examples. In *AAAI conference on Artificial Intelligence*.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. (2017). ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *ACM Workshop on Artificial Intelligence and Security*, pp. 15–26. ACM.

Ciuperca, G., Ridolfi, A., and Idier, J. (2003). Penalized Maximum Likelihood Estimator for Normal Mixtures. *Scandinavian Journal of Statistics 30*, 45–59.

Cook, R. D. (1977). Detection of Influential Observation in Linear Regression. *Technometrics 19*(1), 15–18.

Cuesta-Albertos, J., Gordaliza, A., Matrán, C., et al. (1997). Trimmed k-means: An attempt to robustify quantizers. *The Annals of Statistics 25*(2), 553–576.

Cuesta-Albertos, J. A., Matrán, C., and Mayo-Iscar, A. (2008). Trimming and likelihood: Robust location and dispersion estimation in the elliptical model. *The Annals of Statistics 36*, 2284–2318.

Dalvi, N., Domingos, P., Sanghai, S., Verma, D., et al. (2004). Adversarial classification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 99–108. ACM.

Day, N. E. (1969). Estimating the components of a mixture of normal distributions. *Biometrika 56*(3), 463–474.

De Veaux, R. D. (1989). Mixtures of linear regressions. *Computational Statistics & Data Analysis 8*(3), 227–245.

Deleforge, A., Forbes, F., Ba, S., and Horaud, R. (2015). Hyper-spectral image analysis with partially latent regression and spatial markov dependencies. *IEEE Journal of Selected Topics in Signal Processing 9*(6), 1037–1048.

Deleforge, A., Forbes, F., and Horaud, R. (2015). High-dimensional regression with Gaussian mixtures and partially-latent response variables. *Statistics and Computing 25*(5), 893–911.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological) 39*(1), 1–22.

Duchi, J. C., Jordan, M. I., Wainwright, M. J., and Wibisono, A. (2015). Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory 61*(5), 2788–2806.

Elad, M. and Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing 15*(12), 3736–3745.

Elisseeff, A. and Weston, J. (2002). A kernel method for multi-labelled classification. In *Neural Information Processing Systems*, pp. 681–687.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. (2018). Robust physical-world attacks on deep learning visual classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1625–1634.

Fawcett, T. (2003). In vivo spam filtering: a challenge problem for kdd. *ACM SIGKDD Explorations Newsletter 5*(2), 140–148.

Fawcett, T. and Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery 1*(3), 291–316.

Fitzgerald, D. (2011). Upmixing from mono-A source separation approach. In *International Conference on Digital Signal Processing*, pp. 1–7. IEEE.

Fraley, C. and Raftery, A. E. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association 97*(458), 611–631.

Fraley, C. and Raftery, A. E. (2007). Bayesian regularization for normal mixture estimation and model-based clustering. *Journal of Classification 24*(2), 155–181.

Fritz, H., García-Escudero, L. A., and Mayo-Iscar, A. (2013). A fast algorithm for robust constrained clustering. *Computational Statistics & Data Analysis 61*, 124–136.

Frühwirth-Schnatter, S. (2006). *Finite Mixture and Markov Switching Models*. Springer Science & Business Media.

Gallegos, M. T. and Ritter, G. (2005). A robust method for cluster analysis. *The Annals of Statistics 33*, 347–380.

Gao, X., Jiang, B., and Zhang, S. (2014). On the information-adaptive variants of the admm: an iteration complexity perspective. *Optimization Online 12*, 327–363.

García-Escudero, L. A. and Gordaliza, A. (2007). The importance of the scales in heterogeneous robust clustering. *Computational Statistics & Data Analysis 51*(9), 4403–4412.

García-Escudero, L. A., Gordaliza, A., Greselin, F., Ingrassia, S., and Mayo-Iscar, A. (2017). Robust estimation of mixtures of regressions with random covariates, via trimming and constraints. *Statistics and Computing 27*(2), 377–402.

García-Escudero, L. A., Gordaliza, A., Matrán, C., and Mayo-Iscar, A. (2008). A general trimming approach to robust cluster analysis. *The Annals of Statistics 36*, 1324–1345.

Gershenfeld, N. (1997). Nonlinear inference and cluster-weighted modeling. *The Annals of the New York Academy of Sciences 808*(1), 18–24.

Ghadimi, S. and Lan, G. (2013). Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization 23*(4), 2341–2368.

Goldfeld, S. M. and Quandt, R. E. (1973). A Markov model for switching regressions. *Journal of Econometrics 1*(1), 3–15.

Goldsmith, J., Scheipl, F., Huang, L., Wrobel, J., Gellar, J., Harezlak, J., McLean, M. W., Swihart, B., Xiao, L., Crainiceanu, C., and Reiss, P. T. (2018). *refund: Regression with Functional Data*. R package version 0.1-17.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015a). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015b). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Grais, E. M. and Erdoğan, H. (2012). Hidden markov models as priors for regularized nonnegative matrix factorization in single-channel source separation. In *Annual Conference of the International Speech Communication Association*. ISCA.

Grais, E. M. and Plumbley, M. D. (2017). Single channel audio source separation using convolutional denoising autoencoders. In *IEEE Global Conference on Signal and Information Processing*, pp. 1265–1269. IEEE.

Grais, E. M., Roma, G., Simpson, A. J., and Plumbley, M. D. (2016). Single-channel audio source separation using deep neural network ensembles. In *Audio Engineering Society Convention 140*. Audio Engineering Society.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research 13*(Mar), 723–773.

Grosse, K., Manoharan, P., Papernot, N., Backes, M., and McDaniel, P. (2017). On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*.

Hadi, A. S. and Luceño, A. (1997). Maximum trimmed likelihood estimators: a unified approach, examples, and algorithms. *Computational Statistics & Data Analysis 25*(3), 251–272.

Han, L., Zhang, D., Huang, D., Chang, X., Ren, J., Luo, S., and Han, J. (2017). Self-paced mixture of regressions. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 1816–1822. AAAI Press.

Hathaway, R. J. (1985). A Constrained Formulation of Maximum-Likelihood Estimation for Normal Mixture Distributions. *The Annals of Statistics 13*, 795–800.

Hendrycks, D. and Gimpel, K. (2017). Early methods for detecting adversarial images. In *International Conference on Learning Representations (Workshop Track)*.

Hennig, C. (2000). Identifiablity of models for clusterwise linear regression. *Journal of Classification 17*(2), 273–296.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks 2*(5), 359–366.

Hsu, C.-L. and Jang, J.-S. R. (2010). On the improvement of singing voice separation for monaural recordings using the mir-1k dataset. *IEEE Transactions on Audio, Speech, and Language Processing 18*(2), 310–319.

Huang, D., Han, L., and De la Torre, F. (2017). Soft-margin mixture of regressions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4058–4066. IEEE.

Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., and Tygar, J. (2011). Adversarial machine learning. In *ACM Workshop on Security and Artificial Intelligence*, pp. 43–58. ACM.

Huang, P.-S., Chen, S. D., Smaragdis, P., and Hasegawa-Johnson, M. (2012). Singing-voice separation from monaural recordings using robust principal component analysis. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 57–60. IEEE.

Huang, P.-S., Kim, M., Hasegawa-Johnson, M., and Smaragdis, P. (2014). Singing-voice separation from monaural recordings using deep recurrent neural networks. In *International Society for Music Information Retrieval*, pp. 477–482.

Huang, P.-S., Kim, M., Hasegawa-Johnson, M., and Smaragdis, P. (2015). Joint optimization of masks and deep recurrent neural networks for monaural source separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 23*(12), 2136–2147.

Iizuka, S., Simo-Serra, E., and Ishikawa, H. (2017). Globally and locally consistent image completion. *ACM Transactions on Graphics 36*(4), 107.

Jolliffe, I. T. (1982). A note on the use of principal components in regression. *Journal of the Royal Statistical Society: Series C (Applied Statistics) 31*(3), 300–303.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kotz, S. and Nadarajah, S. (2004). *Multivariate t-distributions and their applications*. Cambridge University Press.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *Technical report, University of Toronto*.

Kurakin, A., Goodfellow, I., and Bengio, S. (2017). Adversarial machine learning at scale. In *International Conference on Learning Representations*.

Lawrence, N. (2005). Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of Machine Learning Research 6*(Nov), 1783–1816.

Lax, P. D. and Terrell, M. S. (2014). *Calculus with applications*. Springer.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature 521*(7553), 436.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*(11), 2278–2324.

Lemasson, B., Pannetier, N., Coquery, N., Boisserand, L. S., Collomb, N., Schuff, N., Moseley, M., Zaharchuk, G., Barbier, E., and Christen, T. (2016). MR vascular fingerprinting in stroke and brain tumors models. *Scientific reports 6*, 37071.

Li, K.-C. (1991). Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association 86*(414), 316–327.

Liu, G., Reda, F. A., Shih, K. J., Wang, T.-C., Tao, A., and Catanzaro, B. (2018). Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision*, pp. 85–100.

Liu, S., Chen, J., Chen, P.-Y., and Hero, A. O. (2018). Zeroth-order online alternating direction method of multipliers: Convergence analysis and applications. In *International Conference on Artificial Intelligence and Statistics*, Volume 84, pp. 288–297.

Liutkus, A., Stöter, F.-R., Rafii, Z., Kitamura, D., Rivet, B., Ito, N., Ono, N., and Fontecave, J. (2017). The 2016 signal separation evaluation campaign. In P. Tichavský, M. Babaie-Zadeh, O. J. Michel, and N. Thirion-Moreau (Eds.), *Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings*, Cham, pp. 323–332. Springer International Publishing.

Ma, D., Gulani, V., Seiberlich, N., Liu, K., Sunshine, J. L., Duerk, J. L., and Griswold, M. A. (2013). Magnetic resonance fingerprinting. *Nature 495*(7440), 187.

Maas, A. L., Le, Q. V., O'Neil, T. M., Vinyals, O., Nguyen, P., and Ng, A. Y. (2012). Recurrent neural networks for noise reduction in robust asr. In *Annual Conference of the International Speech Communication Association*.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Mahoney, M. V. and Chan, P. K. (2002). Learning nonstationary models of normal network traffic for detecting novel attacks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 376–385. ACM.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2009). Online dictionary learning for sparse coding. In *International Conference on Machine Learning*, pp. 689–696. ACM.

Markatou, M. (2000). Mixture models, robustness, and the weighted likelihood methodology. *Biometrics 56*(2), 483–486.

McLachlan, G. and Peel, D. (2000). Mixtures of factor analyzers. In *In Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer.

McLachlan, G. and Peel, D. (2004). *Finite mixture models*. John Wiley & Sons.

McLachlan, G. J., Peel, D., and Bean, R. (2003). Modelling high-dimensional data by mixtures of factor analyzers. *Computational Statistics & Data Analysis 41*(3-4), 379–388.

McLean, M. W., Hooker, G., Staicu, A.-M., Scheipl, F., and Ruppert, D. (2014). Functional generalized additive models. *Journal of Computational and Graphical Statistics 23*(1), 249–269.

Müller, C. H. and Neykov, N. (2003). Breakdown points of trimmed likelihood estimators and related estimators in generalized linear models. *Journal of Statistical Planning and Inference 116*(2), 503–519.

Narayanan, A. and Wang, D. (2013). Ideal ratio mask estimation using deep neural networks for robust speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7092–7096. IEEE.

Nesterov, Y. and Spokoiny, V. (2017). Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics 17*(2), 527–566.

Neykov, N., Filzmoser, P., Dimova, R., and Neytchev, P. (2007). Robust fitting of mixtures using the trimmed likelihood estimator. *Computational Statistics & Data Analysis 52*(1), 299–308.

Nitin Bhagoji, A., He, W., Li, B., and Song, D. (2018). Practical black-box attacks on deep neural networks using efficient query mechanisms. In *Proceedings of the European Conference on Computer Vision*, pp. 154–169.

Papernot, N., McDaniel, P., and Goodfellow, I. (2016). Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519. ACM.

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2536–2544.

Peel, D. and McLachlan, G. J. (2000). Robust mixture modelling using the t distribution. *Statistics and Computing 10*(4), 339–348.

Perthame, E., Forbes, F., and Deleforge, A. (2018). Inverse regression approach to robust nonlinear high-to-low dimensional mapping. *Journal of Multivariate Analysis 163*, 1–14.

Perthame, E., Forbes, F., Deleforge, A., Devijver, E., and Gallopin, M. (2017). *xLLiM: High Dimensional Locally-Linear Mapping*. R package version 2.1.

Qiao, Y. and Minematsu, N. (2009). Mixture of probabilistic linear regressions: A unified view of gmm-based mapping techiques. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3913–3916. IEEE.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association 66*(336), 846–850.

Ravikumar, P., Lafferty, J., Liu, H., and Wasserman, L. (2009). Sparse additive models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 71*(5), 1009–1030.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788.

Ridolfi, A. and Idier, J. (2001). Penalized maximum likelihood estimation for univariate normal mixture distributions. In *AIP Conference Proceedings*, Volume 568, pp. 229–237. AIP.

Rousseeuw, P. J. (1984). Least Median of Squares Regression. *Journal of the American Statistical Association 79*(388), 871–880.

Roy, A., Singha, J., Devi, S. S., and Laskar, R. H. (2016). Impulse noise removal using svm classification based fuzzy filter from gray scale images. *Signal Processing 128*, 262–273.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision 115*(3), 211–252.

Schmidt, M. N. and Olsson, R. K. (2006). Single-channel speech separation using sparse non-negative matrix factorization. In *International Conference on Spoken Language Processing*.

Scrucca, L., Fop, M., Murphy, T. B., and Raftery, A. E. (2017). mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal 8*(1), 205–233.

Simpson, A. J., Roma, G., and Plumbley, M. D. (2015). Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network. In *International Conference on Latent Variable Analysis and Signal Separation*, pp. 429–436. Springer.

Snoussi, H. and Mohammad-Djafari, A. (2002). Penalized maximum likelihood for multivariate Gaussian mixture. In *AIP Conference proceedings*, Volume 617, pp. 36–46. AIP.

Song, D., Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Tramer, F., Prakash, A., and Kohno, T. (2018). Physical adversarial examples for object detectors. In {*USENIX*} *Workshop on Offensive Technologies*.

Song, W., Yao, W., and Xing, Y. (2014). Robust mixture regression model fitting by laplace distribution. *Computational Statistics & Data Analysis 71*, 128–137.

Sprechmann, P., Bronstein, A. M., and Sapiro, G. (2012). Real-time online singing voice separation from monaural recordings using robust low-rank modeling. In *International Society for Music Information Retrieval*, pp. 67–72.

Städler, N., Bühlmann, P., and Van De Geer, S. (2010). ℓ1-penalization for mixture regression models. *TEST 19*(2), 209–256.

Stylianou, Y., Cappé, O., and Moulines, E. (1998). Continuous probabilistic transform for voice conversion. *IEEE Transactions on Speech and Audio Processing 6*(2), 131–142.

Subedi, S., Punzo, A., Ingrassia, S., and McNicholas, P. D. (2013). Clustering and classification via cluster-weighted factor analyzers. *Advances in Data Analysis and Classification 7*(1), 5–40.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *CVPR*, pp. 2818–2826.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science 290*(5500), 2319–2323.

Thayananthan, A., Navaratnam, R., Stenger, B., Torr, P. H., and Cipolla, R. (2006). Multivariate relevance vector machines for tracking. In *European Conference on Computer Vision*, pp. 124–138. Springer.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological) 58*(1), 267–288.

Toda, T., Black, A. W., and Tokuda, K. (2008). Statistical mapping between articulatory movements and acoustic spectrum using a gaussian mixture model. *Speech Communication 50*(3), 215–227.

Tramèr, F., Dupré, P., Rusak, G., Pellegrino, G., and Boneh, D. (2018). Ad-versarial: Defeating perceptual ad-blocking. *arXiv preprint arXiv:1811.03194*.

Turkmen, A. and Billor, N. (2013). Partial least squares classification for high dimensional data using the pcout algorithm. *Computational Statistics 28*(2), 771–788.

Van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Vandev, D. and Neykov, N. (1998). About regression estimators with high breakdown point. *Statistics: A Journal of Theoretical and Applied Statistics 32*(2), 111–129.

Virtanen, T. (2007). Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE Transactions on Audio, Speech, and Language Processing 15*(3), 1066–1074.

Wang, Y., Du, S., Balakrishnan, S., and Singh, A. (2018). Stochastic zeroth-order optimization in high dimensions. In *International Conference on Artificial Intelligence and Statistics*, Volume 84, pp. 1356–1365.

Wang, Z. and Zhang, D. (1999). Progressive switching median filter for the removal of impulse noise from highly corrupted images. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 46*(1), 78–80.

Weninger, F., Hershey, J. R., Le Roux, J., and Schuller, B. (2014). Discriminatively trained recurrent neural networks for single-channel speech separation. In *IEEE Global Conference on Signal and Information Processing*, pp. 577–581. IEEE.

Wu, H.-M. (2008). Kernel sliced inverse regression with applications to classification. *Journal of Computational and Graphical Statistics 17*(3), 590–610.

Xie, B., Pan, W., and Shen, X. (2010). Penalized mixtures of factor analyzers with application to clustering high-dimensional microarray data. *Bioinformatics 26*(4), 501–508.

Xu, L., Jordan, M. I., and Hinton, G. E. (1995). An alternative model for mixtures of experts. In *Advances in Neural Information Processing Systems*, pp. 633–640.

Xu, W., Qi, Y., and Evans, D. (2016). Automatically evading classifiers. In *Network and Distributed Systems Symposium*.

Yang, C., Lu, X., Lin, Z., Shechtman, E., Wang, O., and Li, H. (2017). High-resolution image inpainting using multi-scale neural patch synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6721–6729.

Yang, Y.-H. (2013). Low-rank representation of both singing voice and music accompaniment via learned dictionaries. In *International Society for Music Information Retrieval*, pp. 427–432.

Yao, W., Wei, Y., and Yu, C. (2014). Robust mixture regression using the t-distribution. *Computational Statistics & Data Analysis 71*, 116–127.

Yi, X. and Caramanis, C. (2015). Regularized EM algorithms: a unified framework and statistical guarantees. In *Advances in Neural Information Processing Systems*, pp. 1567–1575.

Zen, H., Tokuda, K., and Black, A. W. (2009). Statistical parametric speech synthesis. *Speech Communication 51*(11), 1039–1064.

Zhao, T., Li, X., Liu, H., and Roeder, K. (2014). *SAM: Sparse Additive Modelling*. R package version 1.0.5.